Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

на тему «Метод побудови архітектури вбудованих систем обробки зображень на основі FPGA»

КвРКІП. 170333.21.01.15 ПЗ

Виконав: студент 2 курсу, група КІ2м-21-1 _____ Ванярха О.С.
Підпис Ініціали, прізвище

Керівник доктор техн. наук, професор _____ Лисенко С.М.
Науковий ступінь, вчене звання Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КІІС, д.т.н., проф.
Т.О. Говорущенко
11 05 2023 р.

Хмельницький, 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень магістр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма освітньо-наукова програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

" 01 " 09 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Ванярхі Олександру Сергійовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Метод побудови архітектури вбудованих систем обробки зображень на основі FPGA

Керівник проекту (роботи) Лисенко С. М., д.т.н., професор

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 09.01.2023 р. № 1

2. Строк подання студентом проекту (роботи) на кафедру 01.05.2023 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Дослідження методу побудови архітектури вбудованих систем обробки зображень на основі FPGA;

Аналіз сучасних програмно-технічних засобів для обробки зображень на основі FPGA;

Розробка моделі функціонування програмно-технічних засобів обробки зображень на основі FPGA;

Розробка методу побудови архітектури вбудованих систем обробки зображень на основі FPGA;

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів кваліфікаційної роботи магістра

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІІС		

7. Дата видачі завдання «_06_»_____09_____2022р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи магістра	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики КвРМ з керівником	05.09.2022	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	05.10.2022	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	05.11.2022	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	05.12.2022	виконано
5	Робота над науковою статтею	05.01.2023	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	15.02.2022	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	05.04.2023	виконано
8	Оформлення пояснювальної записки згідно вимог	15.04.2023	виконано
9	Попередній захист ДРМ	18.04.2023	виконано
10	Захист ДРМ на засіданні ЕК	До 12.05.2023	

Студент _____ Підпис О.С. Ванярха Ініціали, прізвище

Керівник роботи _____ Підпис С.М. Лисенко Ініціали, прізвище

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: метод побудови архітектури вбудованих систем обробки зображень на основі FPGA

Автор роботи: Ванярха Олександр Сергійович

Керівник роботи: Лисенко Сергій Миколайович

Пояснювальна записка: 77 с., 24 рис., 8 табл., 2 дод., 81 джерел.

Ключові слова: FPGA, інструменти високорівневого синтезу, цифрова обробка сигналів, вбудовані системи, системи обробки зображень, паралельні обчислення, архітектури пам’яті, високорівнева обробка зображень.

Об’єктом дослідження є системи обробки зображень на основі FPGA.

Предметом дослідження є модель, метод побудови архітектури вбудованих систем обробки зображень на основі FPGA.

Метою кваліфікаційної роботи магістра є підвищення швидкодії систем обробки зображень на основі FPGA.

Для розв’язання поставлених задач використовувалися основні положення теорії комп’ютерних мереж та систем, системного аналізу, моделювання, методів аналізу даних, теорії математичної статистики, теорії дискретної математики.

Наукова новизна отриманих результатів:

- удосконалено метод побудови архітектури вбудованих систем обробки зображень, який на відміну від відомих використовує FPGA та грунтується на мінімізації потужності, та оптимізованому використанні пам’яті, і який забезпечує високу швидкодію обробки зображень;
- набули подальшого розвитку програмно-технічні засоби обробки зображень на основі FPGA.

На основі проведених досліджень розроблена архітектура і компоненти програмного забезпечення вбудованих систем обробки зображень на основі FPGA.

Практична значимість отриманих результатів полягає у підвищенні швидкодії систем обробки зображень на основі FPGA.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

FPGA - програмована користувачем вентильна матриця

ARM - поліпшена RISC машина

FIFO – першим увійшов, першим вийшов

HLS - інструменти високорівневого синтезу

DSP - цифрова обробка сигналів

BRAM - блокова пам'ять з довільним доступом

ПЛІС - програмована логічна інтегральна схема

GPU – графічний процесор

LUT - таблиця перетворення

ВСТУП

FPGA (Field-Programmable Gate Array) - це електронний пристрій, що містить програмовану логіку та здатний виконувати деякі обчислювальні операції. Це дозволяє збільшувати продуктивність та ефективність роботи комп'ютерних систем, зменшуючи витрати на їх розробку та енергоспоживання.

У дослідженні вивчались різні методи побудови архітектур на основі FPGA, зокрема, моделі потоків даних та паралельні обчислення. Ці методи є особливо корисними для розвитку адаптивних та ефективних апаратних архітектур, оскільки вони забезпечують швидке та точне виконання завдань з обробки даних.

Одним із головних напрямів досліджень є побудова архітектури вбудованих систем обробки зображень, який відрізняється від відомих методів використання FPGA та грунтується на мінімізації потужності оптимізованому використанні пам’яті. Це дозволяє забезпечити високу швидкодію обробки зображень, що є критично важливим для багатьох застосувань, зокрема в медицині, автомобільній та промисловій сферах.

Одним з принципових підходів, що досліджувалися в роботі, є використання потоків даних. Цей підхід базується на тому, що дані обробляються послідовно, як тільки вони стають доступними. Це дозволяє зменшити час виконання операцій та покращити ефективність використання ресурсів FPGA.

Паралельні обчислення є ще одним ефективним підходом, що досліджувався в роботі. Вони дозволяють розділити завдання на багато потоків та обчислювати їх одночасно, зменшуючи час виконання операцій та забезпечуючи швидку обробку великих обсягів даних. Паралельність може бути реалізована на різних рівнях, від рівня бітів до рівня задач.

Метою роботи є підвищення швидкодії систем обробки зображень на основі FPGA.

Для підвищення швидкодії систем обробки зображень на основі FPGA необхідно:

1. дослідити методи побудови архітектури вбудованих систем обробки зображень на основі FPGA;

2. проаналізувати сучасні програмно-технічні засоби для обробки зображень на основі FPGA

3. розробити модель функціонування програмно-технічних засобів обробки зображень на основі FPGA;

4. розробити метод побудови архітектури вбудованих систем обробки зображень на основі FPGA;

5. реалізувати метод побудови архітектури вбудованих систем обробки зображень на основі FPGA.

Об'єкт дослідження – системи обробки зображень на основі FPGA.

Предмет дослідження – модель, метод побудови архітектури вбудованих систем обробки зображень на основі FPGA.

Наукова новизна отриманих результатів:

1. Удосконалено метод побудови архітектури вбудованих систем обробки зображень, який на відміну від відомих використовує FPGA та грунтується на мінімізації потужності оптимізованому використанні пам’яті, і який забезпечує високу швидкодію обробки зображень.

2. Набули подальшого розвитку програмно-технічні засоби обробки зображень на основі FPGA.

Відповідно, можна стверджувати, що вирішення задачі, яке проводиться в даній кваліфікаційній роботі, може мати значний вплив на розвиток адаптивних та ефективних апаратних архітектур вбудованих систем обробки зображень на основі FPGA, що здатні справлятися з обробкою динамічних робочих навантажень даних і в той же час забезпечувати швидкодію. Таким чином, дослідження проведені в роботі можуть внести значний внесок у розвиток комп'ютерних технологій, які використовуються в різних галузях.

1 МЕТОД ПОБУДОВИ АРХІТЕКТУРИ ВБУДОВАНИХ СИСТЕМ ОБРОБКИ ЗОБРАЖЕНЬ НА ОСНОВІ FPGA

1.1 Дослідження паралельних вбудованих архітектур

Протягом останнього десятиліття багатопроцесорні архітектури стали важливою обчислювальною парадигмою для паралельних обчислень [1], [2], [4]. Вони стимулювали розвиток передових паралельних вбудованих архітектур. Тенденція до інтеграції однорідних і різнорідних обчислювальних блоків відкрила різні можливості апаратної декомпозиції, відображення та дослідження паралельних додатків. Апаратні архітектури, що складаються з десятків і сотень легких обчислювальних блоків, стали місцем не тільки для оптимізації продуктивності [2], [3], [5]. У той же час, ці апаратні архітектури створюють ряд проблем, таких як специфічні для архітектури навички перенесення та оптимізації додатків на базову архітектуру, що включає в себе управління та використання паралелізму та гетерогенності системи [5], [7]. Цей розділ охоплює базові дослідження, необхідні для розуміння цих проблем, а також підходи до побудови архітектури вбудованих систем обробки зображень на основі FPGA.

1.1.2 Багатопроцесорна система на основі FPGA

Нові гетерогенні багатопроцесорні системи на чіпові (MPSoC), такі як Altera Cyclone V, мають можливість програмування програмного забезпечення процесора загального призначення (ARM) з апаратною програмованістю FPGA. Інтеграція апаратного та програмного забезпечення зробила архітектуру MPSoC придатною обчислювальною платформою для реалізації змішаної функціональності на одному пристрої, а також для розробки адаптивних вбудованих архітектур [1], [4], [5].

Тим не менш, гетерогенні платформи MPSoC вирішили деякі проблеми апаратного та програмного забезпечення для програмування. Однак, пристосування паралельних обчислювальних задач до базових апаратних ресурсів шляхом використання більшої кількості обчислювальних вузлів, інтегрованих в

один чіп, все ще залишається проблемою. Ця проблема безпосередньо пов'язана з оптимальним пошуком типу та ступеня паралелізму між декількома обчислювальними вузлами, доступними в гетерогенній системі. Крім того, оптимізована реалізація паралельних додатків з використанням цих гетерогенних платформ часто передбачає проектування та розробку апаратних прискорювачів для задоволення вимог додатків. Архітектура цих апаратних прискорювачів може мати динамічний діапазон гнучкості від фіксованої, реконфігурованої, програмно-програмованої або їх комбінації. Вони розміщуються на FPGA і зазвичай управляються процесором загального призначення, наприклад, процесором ARM Cortex-A [7].

1.1.2 Підходи до проектування апаратних прискорювачів на основі FPGA

Silicon vendor та дослідницька спільнота розробили та запропонували різні архітектури, інструменти проектування та програмні фреймворки, які полегшують розробку апаратних прискорювачів. Інструменти Silicon vendor надають згуртоване гетерогенне апаратно-програмне рішення для спільного проектування для розробки та інтеграції апаратних прискорювачів на основі FPGA [4]. Однак, щоб реалізувати інший варіант використання програми, потрібні архітектурні зміни та синтез дизайну. Ці інструменти проектування охоплюють як апаратний, так і програмний простір проектування, який можна розділити на завдання компіляції інтерфейсного програмного забезпечення та завдання компіляції апаратного забезпечення, як показано на рисунку 1.1. Компіляція зовнішнього програмного забезпечення включає опис програми та рівні архітектури прискорювача [9]. Опис програми може бути специфічним для домену чи цілі, тоді як архітектура прискорювача може охоплювати широкий діапазон архітектур апаратного прискорювача. З іншого боку, рівень фізичного відображення використовує інструменти Silicon vendor, щоб фізично відобразити вибрану архітектуру апаратного прискорювача на ресурси FPGA для компіляції внутрішнього обладнання.

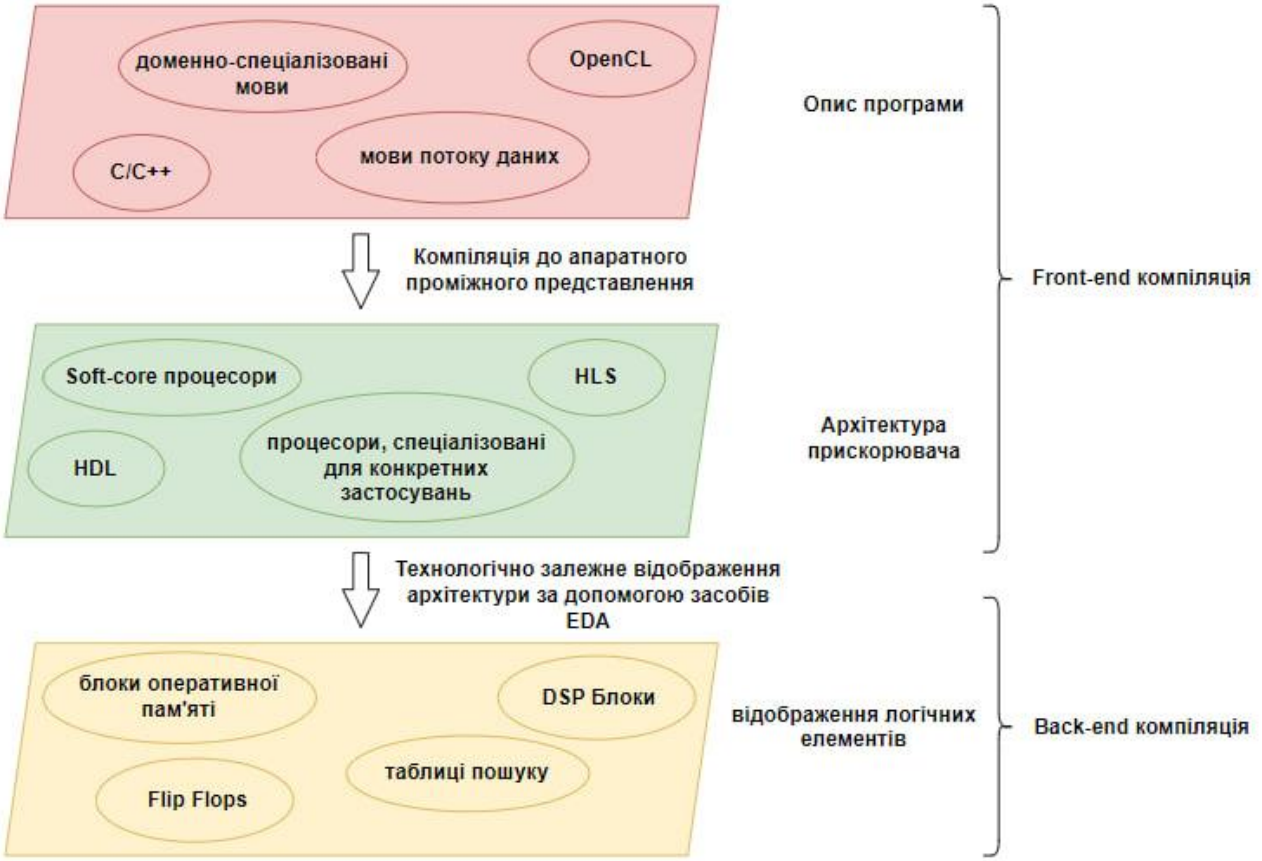


Рисунок 1.1 - Підходи до компіляції дизайну апаратного прискорювача на основі FPGA

Щоб забезпечити абстракцію програмування, додаток можна описати мовою високого рівня, такою як C/C++, OpenCL, або мовою, специфічною для домену. Цей опис програми перекладено, оптимізовано та скомпільовано в проміжне представлення, яке можна відобразити на цільовому апаратному прискорювачі. Можна застосувати широкий спектр підходів цільового апаратного прискорювача, починаючи від високооптимізованого процесора для конкретної програми, гнучкого та програмованого програмного процесора, накладної архітектури або їх комбінації, як показано на рисунку 1.1. Кожен із цих підходів має свої плюси та мінуси щодо гнучкості дизайну, площі та продуктивності. На основі вибраної архітектури цільового апаратного прискорювача проміжне представлення може бути перетворено або в набір спеціальних інструкцій для домену, програмний код, що складається з суміші інструкцій загального призначення, мову опису

обладнання (HDL) або комбінацію з них. Фізичний рівень приймає HDL-опис конструкції цільового апаратного прискорювача та перетворює його у форму FPGA, яка відображається, тобто у фізичні ресурси FPGA (тригери, таблиці пошуку, виділений DSP і блоки пам'яті). Це завдання потребує технологічно залежних механізмів оптимізації та маршрутизації, які виконуються автоматизованими інструментами Silicon vendor. Завдання включають синтез дизайну, генерацію place-and-route та біт-потоку. Ці кроки можуть потребувати значного часу для розробки ітераційного алгоритму залежно від складності та розміру конструкції апаратного забезпечення.

Інструменти високорівневого синтезу (HLS) беруть опис програми, використовують різні методи аналізу для профілювання та дослідження простору дизайну. Більшість цих інструментів використовують модель потоку даних для обчислень, тому на рисунку 1.2 перераховано як академічні, так і комерційні інструменти HLS, які широко описані у відкритій літературі[3],[11]. Ці інструменти підтримують різні мови високого рівня, такі як C, C++, OpenCL або мови домену для опису програми. Ці інструменти профілюють, досліджують, оптимізують і компілюють опис високого рівня в проміжне представлення, яке перекладається на мови опису обладнання, такі як VHDL, Verilog або SystemC, як зазначено у таблиці 1.1. Ці інструменти використовують переваги глибокої конвеєрної обробки FPGA, щоб використовувати паралелізм і досліджувати продуктивність і оптимізацію ресурсів шляхом налаштування розміру FIFO (first-in-first-out) [6]. Завеликий буфер використовує більше ресурсів, ніж потрібно, тоді як малий буфер може спричинити додаткові затримки, зупинки та взаємоблокування під час виконання програми. Проте всі інструменти HLS створюють фіксовану апаратну архітектуру, призначену для прискорення певної програми або частини алгоритму, яка не адаптується. Щоб реалізувати різні програми, єдиною можливістю є переписати та виконати всі зовнішні та внутрішні завдання, описані на рисунку 1.1. Внутрішні завдання можуть значно збільшити час проектування, що не є привабливим для розробників програмного забезпечення та алгоритмів через ітеративний процес

розробки алгоритму, який вимагає дослідження дизайну та швидкого створення прототипів.

Таблиця 1.1 - Інструменти високорівневого синтезу (HLS) для FPGA

HLS Інструмент	Ліцензія	Вхід	Вихід	Потік даних	Потік керуван-ня
Catapult-C	Комерційна	C/C++/ SystemC	VHDL/ Verilog/ SystemC	+	+
Bluespec	Комерційна	BSV	System Verilog	+	+
C-to-Silicon	Комерційна	SystemC/C++	Verilog/ SystemC	+	+
MaxCompile r	Комерційна	MaxJ	RTL	+	-
ROCCC	Комерційна	C Subset	VHDL	+	+
GAUT	Академічна	C/C++	VHDL	+	+
Synphony C	Комерційна	BDL	VHDL/ Verilog	+	+
LegUp	Академічна	C	Verilog	+	+
Vivado HLS	Комерційна	C/C++/ SystemC	VHDL/ Verilog/ SystemC	+	+
Altera SDK	Комерційна	C/Open CL	VHDL/ Verilog	+	+
HIPAcc	Академічна	C++ Embedded DSL	C++	+	-
Merlin Compiler	Комерційна	C/C++	C/OpenCL	+	+

1.1.3 Потреба в адаптивних апаратних архітектурах

Ринки універсальних додатків, що розвиваються, підвищують попит на високопродуктивні та ефективні архітектури на основі FPGA, які можуть обробляти робочі навантаження динамічних даних і в той же час адаптуються для прискорення різних програм.

Один із способів підійти до цієї дослідницької проблеми полягає в розробці адаптивної апаратної архітектури на основі FPGA, яка дозволяє редагувати-компілювати-виконувати потік, знайомий розробникам програмного забезпечення та алгоритмів, замість синтезу апаратного забезпечення та розміщення та маршруту.

Цього можна досягти шляхом заповнення логіки FPGA легкими та високопродуктивними програмними процесорами, які використовуються для програмованого апаратного прискорення.

Ця базова архітектура буде адаптованою та може бути запрограмована за допомогою звичайних підходів до розробки програмного забезпечення, як показано на рисунку 1.1.

Цей підхід не потребує синтезу дизайну апаратного забезпечення, місця та маршруту. Натомість йому знадобиться повторна компіляція програмного забезпечення, яке генеруватиме двійковий код для роботи на базових процесорах з програмним ядром.

Хоча проекти на основі HLS оптимізовані для варіантів використання, оскільки програма відома ще до реалізації основного апаратного забезпечення. Навпаки, у підході, що базується на процесорі, базова апаратна архітектура розроблена, синтезована, розміщено та маршрутизовано заздалегідь.

Таким чином, очікується, що загальна площа буде більш значною, а продуктивність нижчою, ніж HLS, що відбуватиметься за рахунок адаптивності та скорочення часу розробки [8].

Цей підхід забезпечує апаратну абстракцію базових програмованих ресурсів FPGA, дозволяючи їм переконфігурувати за допомогою традиційних програмних

підходів і відкриває це для розробника програмного забезпечення. Він успадковує переваги програмного забезпечення, такі як портативність, спільне проектування апаратно-програмного забезпечення, декомпозицію та параметри відображення для досягнення бажаної області та продуктивності.

Крім того, уникнення необхідного ітераційного процесу синтезу та місця й маршруту зменшить час проектування, підвищить продуктивність і надасть можливість досліджувати проект, керований програмним забезпеченням. Джейн, Рігамонті та Лю повідомили про значне покращення шляхом компіляції додатків на архітектурі процесора через HDL та підходів часткової реконфігурації.

Тим не менш, одним із важливих завдань є ефективна компіляція, відображення та виконання паралельних додатків на основну архітектуру програмованого апаратного прискорювача.

1.1.4 Пам'ять і обчислювальні ресурси FPGA

Структура FPGA забезпечує основні цифрові компоненти, необхідні для створення будь-якої цифрової схеми. Вона має логічні блоки, спеціальну пам'ять і блоки DSP, схеми керування годинником і ресурси маршрутизації для підключення цих цифрових компонентів.

У FPGA розташування цих компонентів є фіксованим і не може бути змінено, тому важливо враховувати їхнє розташування, щоб отримати архітектуру апаратного забезпечення, що економить площу та має високу продуктивність.

У сімействах FPGA Altera Необроблені обчислення (GMAC) прямо пропорційні кількості блоків DSP. Dinechen та ін. [15] показали, як відобразити різні арифметичні оператори на структурі FPGA, використовуючи різні підходи, включаючи LUT, блок DSP тощо. Подібним чином були представлені відображення математичних виразів у цих блоках DSP і досягли покращення продуктивності.

Хоча обчислювальні ресурси та пропускна здатність є високими, пам’ять у FPGA обмежена порівняно з іншими обчислювальними технологіями. На рисунку

1.2 показано розподіл внутрішньої пам'яті та пропускної здатності на FPGA Cyclon V. Віддаляючись від шляху даних, розмір пам’яті збільшується, а пропускна здатність обмежується.

Внутрішня пам’ять складається з розподіленої оперативної пам’яті на основі LUT, яка має невеликі розміри та розташована близько до шляху даних, що може забезпечити швидший доступ до даних із вищою пропускною здатністю. З іншого боку, Block RAM порівняно більший, але обмежений у пропускній здатності. Це показує, що між розміром пам’яті та пропускною здатністю є обмін.

Зосереджуючись на технології FPGA, Altera 5 серії представлені трьома різними сімействами. Серія 5 поєднує в собі процес зниження енергоспоживання, методи проектування та вдосконалення архітектури для забезпечення найнижчого у своєму класі енергоспоживання порівняно з попереднім поколінням Altera FPGA. Він охоплює недороге сімейство Arria, сімейство Straix середнього класу та сімейство FPGA Cyclone високого класу.

Усі три сімейства використовують однакову 28-нанометрову кремнієву технологію та мають базові будівельні блоки FPGA у вигляді логічних комірок, блоків DSP, BlockRAM, що спрощує перенесення дизайнів у сімейство FPGA. Сімейство пристроїв Straix має ідеальний баланс між тактовою частотою інтерфейсу FPGA та енергоспоживанням, високою швидкістю введення/виведення, ємністю та надійністю. Arria використовує ті ж ресурси FPGA, що й Straix, але оптимізовано для ще нижчого енергоспоживання та меншого розміру пакетів, забезпечуючи аналогічні переваги за рахунок нижчої ціни та продуктивності чіпа.

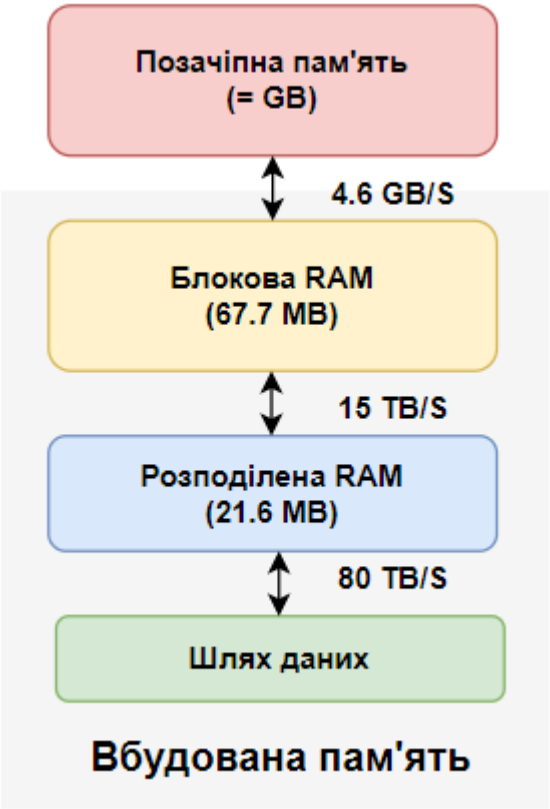


Рисунок 1.2 - Пам'ять FPGA та ієрархія пропускної здатності Altera Cyclone V FPGA.

1.1.5 Блок DSP

Більшість додатків цифрової обробки сигналів широко використовують операції множення та накопичення, які можна ефективно виконувати за допомогою цих блоків DSP.

Ці блоки рівномірно розподілені всередині FPGA. Вони здатні виконувати базові арифметичні та логічні операції над даними, які підходять для розробки ефективного високопродуктивного арифметично-логічного пристрою (ALU) процесора.

Блок DSP Altera (DSP48E1 і DSP48E2) підтримує ці операції та може динамічно налаштовуватися на відміну від Altera.

На рисунку 1.3 показана спрощена функціональна блок-схема DSP48E1. Він має чотири основні арифметичні блоки:

1. 25-бітний попередній суматор;
2. Множник 25x18 біт;
3. 48-бітний суматор, віднімач, логічний;
4. Компаратор і детектор шаблонів.

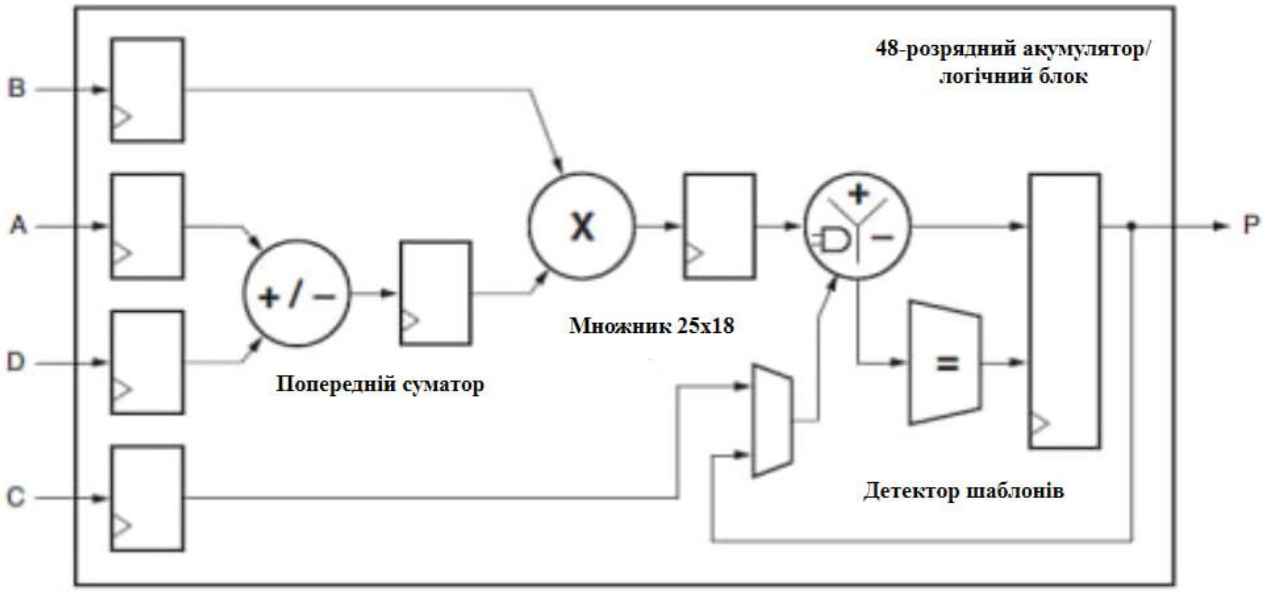


Рисунок 1.3 - Блок-схема виділеного блоку DSP Altera (DSP48E1)

DSP48E1 здатний виконувати операції множення, множення-накопичення, додавання, віднімання та інші операції. Крім того, набір керуючих регістрів, які дозволяють керувати внутрішнім трактом даних від циклу до циклу. Існують конвеєрні регістри, які вмикають/вимикають внутрішню конвеєрну передачу блоку DSP48E1 і покращують синхронізацію блоку шляхом зменшення критичного шляху.

Три внутрішні мультиплексори дозволяють відображати вхідні та вихідні операнди на множник і суматор/віднімач.

1.2 Модель потоку даних обчислень

На початку 1970-х років були введені різні класи моделі обчислень (MoC), які моделюють незалежні від архітектури функціональні вимоги через семантику, інтерфейси та забезпечують синергію між процесорними блоками, вибір відповідного MoC є одним із ключових рішень щодо проектування обладнання. MoC потоку даних може бути експресивним програмуванням і ефективною моделлю виконання. Він має властивість виражати додатки як мережеві процеси, які пропонують паралелізм, масштабованість, модульність, портативність і адаптивність. Ці характеристики життєво важливі для уніфікації дизайну системного рівня гетерогенних платформ. Крім того, він дотримується принципу потокової обробки, який підходить для апаратних архітектур на основі FPGA.

1.2.1 Поняття паралельності в графах потоку даних

Моделі програмування, керовані потоком і потоком даних, дозволяють ефективно реалізувати різні типи паралелізму:

Паралелізм конвеєра. Конвеєр — це ланцюжок акторів a1,...,an, які безпосередньо з’єднані в графі потоку. Кожна пара (ai, ai+1), і 2 {1,...,n 1} має відносини виробник/споживач, тобто ai споживає предмети, вироблені ai1, і виробляє предмети, які служать вхідними даними для ai+1. На рисунку 1.4 показано конвеєрне виконання функцій A і B. Важливо відзначити, що пропускна здатність повинна бути настільки ж швидкою, як і найповільніша група акторів у конвеєрі.

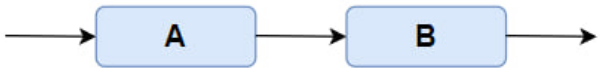


Рисунок 1.4 - Паралелізм конвеєра

Паралелізм завдань. Два актори a1, a2 є паралельними завданнями, якщо вони знаходяться на різних гілках графа потоку. На відміну від конвеєрів, між a1 і a2 немає залежностей вхід/вихід. На рисунку 1.5 показано паралельні актори завдання D і E.

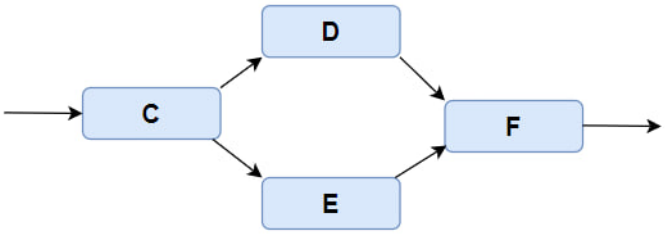


Рисунок 1.5 - Паралелізм завдань

Паралелізм даних — це властивість актора не мати залежностей між одним виконанням і наступним. Актора можна відтворити за допомогою кількох екземплярів актора, наприклад G замінюється двічі, як показано на рисунку 1.6.

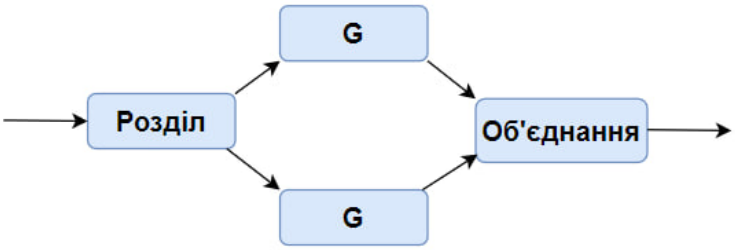


Рисунок 1.6 - Паралелізм даних

1.2.2 Перетворення потоку даних

Перетворення потоку даних часто використовуються для підвищення продуктивності системи шляхом покращення продуктивності повільніших вузлів потоку даних або частини графа. Ці перетворення зберігають функціональність початкового графіка потоку даних, але збільшують пропускну здатність або

зменшують затримку. Графи потоку даних піддаються грубому перетворенню для використання паралелізму даних, завдань і конвеєрів, який можна ефективно реалізувати за допомогою FPGA. Апаратна архітектура на основі однієї інструкції з кількома даними (SIMD) використовувалася для прискорення додатків, включаючи попередню обробку зображень завдяки масовій обробці пікселів. Для покращення продуктивності можна використовувати спеціальну оптимізацію потоку даних (декомпозиція, відображення та планування) і перетворення (поділ, злиття тощо). Ці перетворення дозволяють декомпозицію та можливості розробки простору для досягнення бажаних цілей застосування. Програма може відображатися на багатоядерній архітектурі, що дозволить використовувати паралелізм даних і завдань, підтримуючи потік проектування «редагування-компіляція-запуск».

1.3 Шаблони паралельних обчислень

Шаблони паралельних обчислень охоплюють загальні парадигми паралельного програмування та абстрагуються для програміста як конструкції програмування високого рівня, оснащені чітко визначеною функціональною семантикою. Вони моделюють точний паралельний шаблон для використання паралелізму та приховують деталі реалізації шаблону від програміста для використання паралелізму, як показано на рисунку 1.7. Ці шаблони є параметричними і можуть повторно використовуватися в різних програмах. Цей підхід використовується в кількох структурах паралельного програмування.

1.3.1 Конвеєр

Основна ідея шаблону конвеєра полягає в тому, щоб розділити обробку на серію послідовних кроків із збереженням у кінці кожного кроку. Це можливо шляхом розподілу послідовної програми на кілька незалежних, але послідовних завдань, де попереднє завдання передає дані наступному. Це забезпечує

паралельність, коли ці завдання можуть виконуватися паралельно, щойно дані стають доступними на вузлі обробки. Обчислювальне навантаження завдань може змінюватись і невідоме до виконання, якщо для визначення паралельної програми не використовується статична модель обчислень, наприклад статичний потік даних. Однак максимально досяжна швидкість обробки залежить від швидкості обробки найповільнішого завдання, яка є швидшою за час, необхідний для виконання всіх кроків одночасно. Однак за допомогою статичного профілювання наявної програми можна знайти ефективну декомпозицію, яка може призвести до збалансованих завдань з обмеженими вимогами до пам’яті.

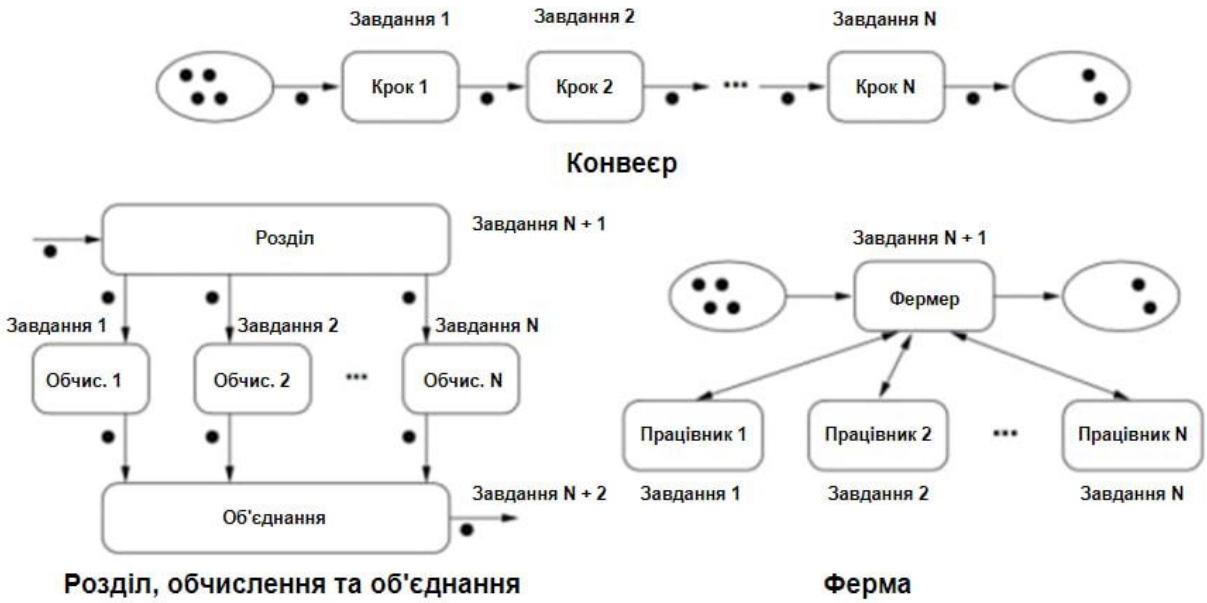


Рисунок 1.7– Схеми паралельних обчислень з використанням акторів потоку даних

1.3.2 Розділ, обчислення та об’єднання

Цей шаблон використовується для обробки регулярно розподілених даних на основі статичного розкладання. Дані поділяються на кілька блоків однакового розміру (на основі рядків, стовпців або блоків), де кількість паралельних блоків даних визначає рівень паралелізму даних, який можна використовувати. З точки

зору архітектури, це відоме як модель паралельного програмування розсіяного збору або розділеного обчислення та злиття. Крім того, його також можна розширити для реалізації різних похідних багатоступеневих конвеєрних шаблонів, щоб використовувати паралелізм як даних, так і завдань, використовуючи конвеєр, розділення, обчислення, зв'язок, обчислення, злиття або їх поєднання для досягнення кращої продуктивності. Перевага конвеєрної обробки кількох етапів полягає в тому, що вона зменшує накладні витрати на передачу даних, покращує пропускну здатність даних, уникає вузьких місць пам'яті на відміну від підходу прискорення на основі моделі спільної пам'яті, де пропускна здатність і когерентність кешу значно погіршують продуктивність.

1.3.3 Ферма

Цей шаблон використовується для обробки нерегулярних даних. Фермер (хост/головний процесор) розподіляє завдання між працівниками, поки не залишиться жодного. Потім фермер чекає результату від працівника і негайно надсилає йому інший робочий елемент. Кожен працівник отримує робочий пакет, обробляє його та повертає результат фермеру, доки він не отримає умову зупинки від фермера. Перевага цього підходу полягає в тому, що фермер знає, які працівники принесли результати своїх завдань і, отже, простоюють. Таким чином, фермер може пересилати вхідні завдання непрацюючим працівникам.

Однак такий підхід має свої недоліки. Це викликає значні накладні витрати через обмін повідомленнями між фермером і робітниками. Крім того, фермер може стати вузьким місцем, якщо кількість працівників велика. У цьому випадку фермер не зможе зайняти всіх працівників, що призведе до втрати робочої сили. Кількість робітників, яких може зайняти фермер, залежить від обсягу завдань і обсягу повідомлень, які фермер повинен поширювати. Однак, оскільки процес призначення завдань є циклічним, це може призвести до тупикових блокувань у випадку завдань, що залежать від даних, де обчислення певних робітників може залежати від результатів інших, що призводить до тупикових блокувань. З іншого

боку, потрібно подбати про те, щоб фермер якомога швидше реагував на нові завдання та робітників, які досягли результатів.

Оскільки розробники додатків стикаються з труднощами, ефективно використовують доступні ресурси без знання апаратного забезпечення. Він передбачає обробку низькорівневого ядра, міжядерного та системного зв'язку та системних інтерфейсів тощо.

1.4 Пов'язана робота над програмними процесорами FPGA

Для оцінки цих процесорів акцент буде зроблено на різних розмірах слів, максимальних тактових частотах і використанні ресурсів. Розмір слова є важливим параметром, оскільки для точного представлення піксельних даних у різних колірних просторах з деякою надмірністю потрібні принаймні 16-бітні слова. Тактова частота безпосередньо впливає на максимальну пропускну здатність конструкції, що, у свою чергу, впливає на спостережуване прискорення. Процесори з меншим використанням ресурсів дозволяють використовувати більше логіки для багатоядерних архітектур, що може досягти чудової продуктивності.

Комерційними готовими пропозиціями від провідних постачальників FPGA Xilinx і Altera є процесори MicroBlaze і Nios II відповідно. Обидва є 32-розрядними програмними процесорами на основі RISC-архітектури, які супроводжуються відповідними ланцюжками інструментів розробки програмного забезпечення. Оптимізована продуктивність MicroBlaze здатна забезпечувати до 262 DMIP з 5-ступеневим конвеєром, тоді як Altera здатна забезпечувати 30 DMIPS. Як Nios II, так і MicroBlaze мають широкі можливості конфігурації з такими опціями, як блок з плаваючою комою, керування пам'яттю та підключення до спеціальних апаратних прискорювачів. Ці комерційні програмні процесори були досліджені та модифіковані в кількох статтях. Але, вдалося досягти максимальної робочої частоти в межах 77 - 112 МГц.

Інші процесори доступні за ліцензіями з відкритим кодом, наприклад OpenRISC. Це процесор із відкритим вихідним кодом на основі RISC із 32- та 64-

розрядними режимами та додатковою підтримкою векторів. LEON3 — це 32-розрядний процесор, сумісний із SPARC V8, описаний у VHDL, який доступний під GNU GPL. Він використовує 7-ступінчастий конвеєр, включає блок з плаваючою комою, підтримує симетричну багатопроцесорність і працює на частоті до 125 МГц.

У літературі було представлено багато архітектур багатоядерних процесорів FPGA для прискорення різних програм. Архітектура прискорювача Silicon Hive замінила прискорювачі ASIC на реконфігуровані ядра, зробивши прискорювачі повністю програмованими після виготовлення та гнучкими для обслуговування протягом життєвого циклу продукту. Основним компонентом архітектури Silicon Hive є елемент обробки та зберігання (PSE), який складається з кількох функціональних блоків (FU), з'єднаних через мережу з'єднань (IN). Він має один або кілька слотів для проблемних операцій (IS), пов'язаних із FU, файлами розподілених реєстрів (RF) і додатковим локальним сховищем пам'яті (MEM). PSE розроблено таким чином, щоб забезпечити прості та чисті шляхи даних для обробки компілятором і гарантувати високий рівень програмованості. Матриця з одного або кількох PSE разом із контролером (CTRL) і конфігураційною пам'яттю (CONFIG. MEM) становить комірку. PSE в клітині можуть спілкуватися один з одним через лінії передачі даних (CL). Масив з однієї або кількох комірок, з'єднаних через керований даними механізм зв'язку, утворює потоковий масив. Зв'язок між осередками відбувається через блокування FIFO, доступ до якого здійснюється з блоків завантаження/збереження (LD/ST) у осередках, що дозволяє одночасно відображати декілька функцій на потоковому масиві. Ден та ін. Розширив підхід Silicon Hive і запропонував процесор HiveFlex Moustique-IC2 як синтезоване ядро soft-RTL із підсистемою введення/виведення, спеціально розробленою для програм обробки зображень. Moustique-IC2 була машиною SIMD (Single-InstructionMultiple-Data), що означає, що та сама програма одночасно працює з усіма пікселями. Збільшуючи коефіцієнт SIMD, ту ж програму можна використовувати для обробки більшої кількості пікселів одночасно, тим самим

збільшуючи пропускну здатність. 24-канальний процесор SIMD досяг робочої частоти 200 МГц за 90 нм технологією.

Duller та ін. [14] запропонували PicoArray, яка є масивною паралельною архітектурою, розробленою як альтернатива для створення дизайнів ASIC, які є складними для проектування, дорогими за вартістю та потребують більше часу та зусиль на проектування. PicoArray — це мозаїчна архітектура процесора, що складається з великої кількості різнорідних процесорних ядер. Архітектура була в основному розроблена для додатків бездротової інфраструктури. Процесори організовані у двовимірну сітку та з'єднані між собою за допомогою детермінованого з'єднання picoBus. Протокол міжпроцесорного зв'язку ґрунтувався на схемі мультиплексування з тимчасовим розподілом (TDM), де передача даних між портами процесора відбувається протягом автоматично запланованих часових інтервалів інструментом і контрольованих комутаторами шини. Зв'язок між процесорами фіксується під час компіляції і не може бути змінений динамічно. PicoArray розроблено як 16-розрядний 3-процесорний процесор VLIW RISC із гарвардською архітектурою пам'яті. Він підтримує чотири різні варіанти процесорів (стандартний, множення-накопичення, пам'ять і керування). Кожен варіант був розроблений для поєднання DSP, потокової та блочної обробки і, отже, мав різний розподіл внутрішньої пам'яті. Усі чотири варіанти використовують той самий набір інструкцій RISC, за винятком інструкцій MAC, які можуть бути виконані лише на стандартному процесорі. За винятком завантажень і розгалужень, усі інструкції виконуються в одному циклі. Кожен процесор може отримати доступ лише до своєї внутрішньої пам'яті (від 1 КБ до 32 КБ) і спілкуватися з іншими процесорами за допомогою портів введення/виведення даних. Кожен процесор ініціалізувався за допомогою спеціальної шини конфігурації та програмувався за допомогою мови асемблера. PicoChip PC102 працює на частоті 160 МГц на Altera Cyclone V.

Класична векторна обробка передбачає надсилання потоку значень у конвеєрні функціональні блоки. Пізніше було запропоновано кілька архітектур. Було зроблено деякі оптимізації, щоб пришвидшити продуктивність і зменшити час

виконання шляхом включення векторного ланцюжка, виконання потоку керування та файлу банківських реєстрів тощо. Процесор працює на максимальній робочій частоті 200 МГц. Інші запропонували багатопроцесорні архітектури для використання прихованого паралелізму в деяких частинах потокових програм для ефективної реалізації, наприклад VENICE та VectorBlox MXP розроблені для використання паралелізму рівня даних (DLP) шляхом обробки векторів. Також було запропоновано спеціальний програмний процесор GraphSoC для прискорення графових алгоритмів. Це 3-ступеневий конвеєрний процесор, який підтримує семантику графа (операції вузлів, країв). Одна FPGA може вмістити кілька екземплярів цих процесорів, з'єднаних між собою за допомогою мережі на чіпові (NoC). Функціональний опис графіків зберігається у вбудованій BRAM для швидкого локального доступу. Більші графіки можна розділити на підграфи та завантажувати один за одним або розділити на кілька процесорів. Виконувальний етап процесора можна налаштувати та підтримує чотири інструкції для графа, тобто (надсилати, отримувати, накопичувати та оновлювати), які реалізовані як мікрокодований шлях даних. Натомість шлях даних процесора не має файлу регістрів, він має регістри спеціального призначення для зберігання інформації про край і вузол. Повідомлені результати синхронізації показують, що повністю конвеєрна конструкція може працювати на частоті 200 МГц.

FlexGrip — це 32-розрядна багатоядерна масштабована настроювана архітектура процесора, заснована на однопотоковій моделі (SIMT), у якій інструкція отримується та відображається одночасно на кількох скалярних процесорах (SP). Потоковий мультипроцесор (SM) складається з кількох SP, які забезпечують багатопотокове виконання. Кількість потоків еквівалентна кількості скалярних процесорів у потоковому мультипроцесорі (SM). SM — це п'ятиетапна конвеєрна архітектура, яка складається з етапів вибірки, декодування, читання, виконання та запису. Етап виконання складається з кількох скалярних процесорів і одного блоку потоку керування. Цей пристрій працює за інструкціями потоку керування, такими як інструкції розгалуження та синхронізації. Кожен потік відображається в одному скалярному процесорі, що забезпечує паралельне

виконання потоків. Етап запису зберігає проміжні дані у файлі векторного регістру, адреси пам'яті у файлі регістра адреси та прапори предикатів у файлі регістра предикатів. Кінцеві результати зберігаються в глобальній пам'яті. Конструкція з одним SM і 8 SP, реалізована на пристрої Altera Cyclone V, досягла максимальної робочої частоти 100 МГц.

Після огляду різних архітектур процесорів з м’яким ядром головним архітектурним завданням для розробки легкої архітектури процесора з м’яким ядром є пошук найкращого компромісу між функціональністю (здатністю виконувати різні ядра) та продуктивністю (здатністю відповідати вимогам до продуктивності). Тому що вибір дизайну, зроблений на рівні програмного ядра, визначає функціональність і продуктивність багатоядерної платформи та платформи апаратного прискорення.

1.5 Висновки

Робота присвячена удосконаленню методу побудови архітектури вбудованих систем обробки зображень з використанням FPGA. У ході роботи буде запропоновано метод, який відрізняється від відомих тим, що грунтується на мінімізації споживання енергії та оптимізованому використанні пам'яті.

Основна ідея методу полягає у використанні FPGA, що дозволяє ефективно реалізувати обробку зображень з високою швидкодією та мінімальним споживанням енергії. Для досягнення цієї мети було розроблено алгоритм, який дозволяє мінімізувати споживання енергії та оптимізувати використання пам'яті вбудованої системи. За результатами експериментів, буде показано, що запропонований метод забезпечує високу швидкодію обробки зображень порівняно з існуючими методами, а також має вищу ефективність за рахунок мінімізації споживання енергії та оптимізації використання пам'яті.

Запропонований метод побудови архітектури вбудованих систем обробки зображень на базі FPGA є ефективним і може бути використаний для створення потужних вбудованих систем обробки зображень з мінімальним споживанням

енергії та високою продуктивністю. Дослідження має великий потенціал для використання в різних областях, що потребують обробки зображень з високою швидкодією та низьким споживанням енергії.

Поставлена мета досягається розв’язанням таких основних задач:

1. дослідити методи побудови архітектури вбудованих систем обробки зображень на основі FPGA;
2. проаналізувати сучасні програмно-технічні засоби для обробки зображень на основі FPGA;
3. розробити модель функціонування програмно-технічних засобів обробки зображень на основі FPGA;
4. розробити метод побудови архітектури вбудованих систем обробки зображень на основі FPGA;
5. реалізувати метод побудови архітектури вбудованих систем обробки зображень на основі FPGA.

2 МОДЕЛЬ ФУНКЦІОНУВАННЯ СИСТЕМИ ОБРОБКИ ЗОБРАЖЕНЬ НА ОСНОВІ FPGA

2.1 Модель системи обробки зображень на основі FPGA

Удосконалення технології FPGA (Field Programmable Gate Array) [8] зробили їх де-факто платформою впровадження для різноманітних програм комп’ютерного зору [9].

Кілька алгоритмів, наприклад, стереозбіг [8], неможливо обробляти в режимі реального часу на звичайних процесорах загального призначення, і вони найкраще підходять для апаратної реалізації [11].

Відсутність достатньо всеосяжного, універсального апаратного конвеєра для домену комп’ютерного бачення [13] спонукає до використання FPGA у безлічі сценаріїв комп’ютерного бачення, особливо в програмах, де обробка повинна виконуватися на місці, наприклад, у смарт-камерах [15], де FPGA вбудовують підсистеми збору даних, обробки та зв’язку.

Прийняття технології FPGA спільнотою комп’ютерного бачення прискорилося протягом останніх років завдяки наявності інструментів високорівневого синтезу (HLS), які дозволяють проектувати FPGA у встановлених контекстах розробки програмного забезпечення.

Це дає змогу забезпечувати високу продуктивність та ефективність обробки даних у складних системах, що раніше було не досяжним для звичайних процесорів загального призначення.

Застосування FPGA у сучасних системах комп'ютерного бачення також забезпечує високу точність обробки даних, що є критичним для деяких застосувань, наприклад, у медицині або в безпекових системах [18].

HLS дозволяють програмістам розробляти програми для FPGA на високорівневому рівні, застосовуючи мови програмування високого рівня, такі як C++ або Python, і трансформуючи ці програми в апаратний код, який може бути виконаний на FPGA [16].

Це дозволяє програмістам зосередитися на розробці алгоритмів, не

звертаючи уваги на нижчі рівні апаратури, такі як пропускна здатність і інші аспекти апаратного дизайну. Крім того, це зменшує час розробки та підтримки коду, що розгортається на FPGA.

Загалом, FPGA дозволяють виконувати обчислення швидше, ніж звичайні процесори, а також дозволяють виконувати обчислення в режимі реального часу для великої кількості застосувань комп'ютерного бачення.

Це робить їх де-факто платформою впровадження для багатьох програм комп'ютерного зору, зокрема для стереозбігу, обробки зображень та машинного навчання.

З розвитком інструментів високорівневого синтезу, таких як HLS, використання FPGA для виконання завдань комп'ютерного бачення стає все більш доступним і простим для програмістів.

Для досягнення цілей, описаних вище, підхід обраний для вирішення задачі до розподілу ресурсів пам'яті на чіпові базується на детальному формальному аналізі схем розподілу пам'яті та їх пов'язаного використання для заданих розмірів кадрів і можливих конфігурацій пам'яті на чіпові.

На основі цього аналізу, запропоновані методи вибору конфігурації пам'яті для оптимізованого використання ресурсів пам'яті на чіпові та збалансованого використання/потужності для певного розміру кадру.

Крім того, було проведено теоретичний аналіз впливу даних методів розділення на використання ресурсів і енергоспоживання.

Цей аналіз дає змогу зрозуміти, як ці методи можуть покращити енергоефективність і продуктивність FPGA, зменшуючи споживання енергії та збільшуючи продуктивність обробки зображень на FPGA.

Нарешті, здійснюється емпірична перевірка використання ресурсів, потужності та продуктивності запропонованих методів порівнянно з комерційним інструментом HLS.

У вирішенні задачі використовується широкий спектр тестових зображень різних розмірів та складності, щоб зрозуміти, як обрані методи впливають на продуктивність та енергоефективність FPGA.

Цей метод розподілу ресурсів пам'яті на чіпові дозволяє зменшити використання ресурсів та енергоспоживання, покращити продуктивність та енергоефективність FPGA для обробки зображень, та зменшити загальні витрати на FPGA.

Нарешті, ці методи можуть бути застосовані до будь-яких високорівневих систем обробки зображень, таких як обробка відео, розпізнавання обличь, обробка медичних зображень тощо.

Розподіл ресурсів пам'яті на чіпові може допомогти зменшити залежність від зовнішньої пам'яті та знизити енергоспоживання, що є важливим у сучасних високоефективних системах.

Більше того, ці методи можуть бути корисні при розробці систем з обмеженими ресурсами, такими як вбудовані системи, де розмір пам'яті є обмеженим.

Щоб підсумувати, розподіл ресурсів пам'яті на чіпові є важливим аспектом проектування FPGA-систем для обробки зображень.

Використання цих методів може допомогти зменшити використання ресурсів та енергоспоживання, збільшити продуктивність та знизити залежність від зовнішньої пам'яті. Для реалізації цих методів можна використовувати різноманітні інструменти, такі як HLS та HDL, та застосовувати їх до будь-яких високорівневих систем обробки зображень.

Проведені в роботі дослідження показали, що зменшення динамічного енергоспоживання внутрішньої пам'яті може бути досягнуто зменшенням напруги живлення та оптимізацією алгоритмів доступу до пам'яті.

Це дає можливість знизити загальну потужність системи на 70%, що може бути великою перевагою в багатьох сферах.

Для досягнення цієї мети використовуються репрезентативні високорівневі алгоритми, які дозволяють підтримувати продуктивність на потрібному рівні.

Кваліфікаційна робота також включає дослідження оптимізації потужності в рамках архітектури систем пам'яті FPGA та дизайну для обробки зображень.

Одна з ключових проблем, з якими довелось зіткнутись, полягає в

обмеженнях стандартних підходів HLS, що змушує шукати альтернативні методи розподілу. У дослідженні були розглянуті різні способи розподілу для досягнення оптимальної потужності та продуктивності.

В ході роботи використовується термін BRAM (блокова пам'ять з довільним доступом), номенклатуру Altera для вбудованої пам'яті, для позначення пам'яті FPGA на чіпові в цілому.

Також детально описується експериментальну методологію та експериментальні результати, що дає змогу зробити висновки про ефективність запропонованих методів.

Також, стосовно використання терміну BRAM, варто зазначити, що це поняття є досить поширеним в контексті FPGA, і відповідає блоковій пам'яті з довільним доступом.

У Altera цей термін використовується для позначення вбудованої пам'яті на чіпі FPGA в цілому, а не лише блокової пам'яті.

За допомогою BRAM можна зберігати дані, що використовуються у процесі обчислень, що є дуже важливим для багатьох застосувань FPGA.

Остаточні результати дослідження, яке буде описано в даній роботі, мають великий потенціал для практичного використання в різних сферах, таких як обробка зображень, відео аналітика, машинне навчання.

Оскільки питання оптимізації потужності є актуальним і складним, дана робота може стати важливим кроком в розробці більш ефективних алгоритмів та методів оптимізації використання ресурсів FPGA.

2.2 Обгрунтування вибору моделі

У підсистемах обробки FPGA алгоритми еволюціонують від типових представлень, придатних для програмного забезпечення, до більш дружніх до апаратного забезпечення [15], які можуть повністю використовувати паралелізм даних [16] через специфічну для додатків апаратну архітектуру [15], яка часто суттєво відрізняється від традиційної моделі фон Неймана, наприклад потік даних

[14] або біологічна обробка [17].

Ці різнорідні архітектури налаштовані для реалізації FPGA з різних причин, а саме для досягнення більшої продуктивності та енергоефективності. Наприклад, шляхом використання двійкової логарифмічної арифметики для ефективного множення/ділення [16], можна забезпечити більш швидку обробку даних.

Крім того, статичне/динамічне масштабування частоти в паралельних трактах даних допомагає зменшити енергоспоживання.

Найчастіше програми комп'ютерного бачення, розгорнуті на FPGA, обмежені вимогами до продуктивності апаратури, потужності та реального часу [18].

Програми потокового передавання в реальному часі (тобто виконання обробки зображень у каналах відео в реальному часі) вимагають обмеженого часу отримання, обробки та зв'язку, що може бути досягнуто, зберігаючи необхідну обчислювальну потужність, лише шляхом використання паралелізму даних [16] спеціальними засобами функціональних блоків.

Окрім програм комп'ютерного бачення та програм потокового передавання, FPGA також може бути використана в інших додатках, таких як криптографічні системи та схеми зв'язку.

Криптографічні системи можуть вимагати великої кількості обчислень, таких як шифрування та розшифрування, що можуть бути здійснені з високою швидкістю за допомогою FPGA [19].

Схеми зв'язку також можуть бути реалізовані на FPGA, дозволяючи отримати високу пропускну здатність та ефективність використання енергії.

Узагальнюючи, можна сказати, що FPGA забезпечує гнучкість та продуктивність в розробці спеціалізованих обчислювальних систем з урахуванням вимог до продуктивності та енергоефективності.

FPGA дозволяє реалізувати специфічні алгоритми та архітектури, що можуть бути оптимізовані для конкретних завдань та додатків.

Це робить FPGA корисним інструментом для високопродуктивних обчислювальних систем у багатьох галузях, включаючи науку, медицину, фінанси

та технології передачі даних.

Навіть із зростанням пам'яті, досягнення виконання глобальних операцій на FPGA залишається складною задачею. Один із можливих рішень полягає у використанні стратегії динамічного управління пам'яттю, такої як кешування [20].

Це дозволяє зберігати лише необхідні дані в обмеженому обсязі пам'яті, що підвищує ефективність виконання глобальних операцій та зменшує витрати на обробку даних.

Однак, кешування не завжди є ефективним рішенням для обробки зображень, особливо якщо дані мають великі розміри або їх обробка вимагає великої кількості глобальних операцій.

В такому випадку, можливим рішенням є використання декількох FPGA в паралельному режимі, що дозволяє розподілити завдання обробки між кількома пристроями.

Окрім того, існують деякі альтернативні рішення для обробки зображень, які не вимагають використання FPGA, наприклад, графічні процесори (GPU) [21].

GPU можуть бути більш ефективними для обробки зображень, які вимагають використання глобальних операцій, завдяки їх великій кількості обчислювальних ядер та підтримці спеціалізованих операцій зображення.

У будь-якому випадку, FPGA залишаються потужним інструментом для обробки зображень з великою кількістю перспективних можливостей, які можуть бути використані для вирішення складних завдань обробки зображень у майбутньому.

Додатковим підходом до поліпшення використання FPGA для обробки зображень є використання спеціально розроблених алгоритмів стиснення зображення, таких як JPEG або MPEG [21].

Ці алгоритми можуть дозволити зменшення розміру зображення до певного рівня без втрати якості, що дозволить більш ефективно використовувати пам'ять на FPGA.

Однак, такі алгоритми мають свої обмеження, оскільки вони можуть призвести до зниження якості обробки зображень, особливо якщо

використовується багато шарів стиснення.

Також важливо враховувати, що розмір кадрів може змінюватися в залежності від конкретного додатку, тому потрібні методи динамічної оптимізації розміру кадрів, щоб забезпечити ефективну роботу FPGA на різних масштабах зображень.

Крім того, існують інші підходи до вирішення проблеми пам'яті на FPGA для обробки зображень, такі як використання зовнішньої пам'яті або розподіл пам'яті між кількома FPGA [22].

Однак, ці підходи можуть бути складними в реалізації та вимагати більшої кількості обчислювальних ресурсів.

Таким чином, хоча пам'ять є важливим фактором обмеження для широкого використання FPGA для складних програм обробки зображень, існують різні підходи до вирішення цієї проблеми, включаючи вдосконалення алгоритмів обробки зображень, використання спеціальних алгоритмів стиснення зображення, а також використання зовнішньої пам'яті та розподіл пам'яті між кількома FPGA.

Крім того, існують інші напрями дослідження в галузі використання FPGA для обробки зображень. Один з таких напрямів - розробка нових алгоритмів обробки зображень, які більш ефективно працюють на FPGA.

Наприклад, можна використовувати алгоритми зі зменшеною складністю, які виконують тільки ті операції, які необхідні для конкретної задачі обробки зображень.

Це дозволить зменшити кількість ресурсів FPGA, необхідних для обробки зображень, і підвищити продуктивність системи.

Також важливим аспектом є розробка нових архітектур FPGA для обробки зображень. Нові архітектури можуть бути спрямовані на покращення певних аспектів обробки зображень, таких як швидкість обробки, енергоефективність та точність.

Наприклад, можна розробити архітектури, які підтримують паралельну обробку зображень на різних рівнях абстракції, що дозволить підвищити продуктивність системи.

Крім того, важливим напрямом дослідження є розробка нових інтерфейсів для взаємодії між FPGA та позакристальною пам’яттю, таких як високошвидкісний інтерфейс PCIe.

Це дозволить збільшити швидкість передачі даних між FPGA та зовнішнім пристроєм, що може бути особливо важливим для додатків обробки відео інформації.

Для розширення можливостей вбудованої пам'яті FPGA для додатків обробки зображень можуть бути використані різноманітні підходи, такі як розробка нових алгоритмів обробки зображень, що працюють на менших розмірах кадрів, що містяться на FPGA, або застосування інтелектуальних схем розподілу пам'яті на чіпові для розміщення повних кадрів з урахуванням профілів потужності.

Ще одним підходом є рефакторинг алгоритму потокової обробки для мінімізації вимог до пам'яті, що дозволяє ефективніше використовувати вбудовану пам'ять.

Також можна використовувати абстракції мови програмування для ефективної генерації апаратного конвеєра.

Окрім того, можуть бути застосовані методи оптимізації конфігурацій внутрішньої пам'яті чіпу з метою максимально ефективного використання, хоча це часто суперечить орієнтованим на продуктивність схемам розподілу, стандартним у генераторах коду HLS.

2.3 Архітектури пам'яті для обробки даних зі спеціальним призначенням

Існує багато інших підходів до проектування архітектур пам'яті для розподіленої обробки даних, які використовуються в різних сферах, включаючи комп'ютерний зір і машинне навчання.

Наприклад, інші дослідники розробили архітектури пам'яті з використанням спільної пам'яті (shared memory), що дозволяє більш ефективно використовувати ресурси і зменшити кількість дублювання даних.

Крім того, було запропоновано архітектури пам'яті, що використовують

генеративні моделі, які відображають відносини між даними, для зберігання і пошуку зображень.

Одним з основних викликів при проектуванні архітектур пам'яті для розподіленої обробки даних є балансування потреб ресурсів, швидкості та точності.

Наприклад, деякі архітектури пам'яті можуть забезпечувати швидкий доступ до даних, але за рахунок погіршення точності обчислень.

Інші можуть забезпечувати високу точність обчислень, але за рахунок затримки обробки даних.

При проектуванні архітектур пам'яті, необхідно враховувати специфіку застосування та потреби конкретного додатку, щоб забезпечити оптимальний баланс між цими параметрами.

У додатках комп'ютерного зору, таких як відеоспостереження та розпізнавання облич, архітектури пам'яті відіграють важливу роль у підвищенні продуктивності та ефективності роботи системи.

У машинному навчанні, архітектури пам'яті допомагають зберігати великі об'єми даних.

Для розв'язання цих проблем відомі різні підходи, одним із яких є використання розподіленої пам'яті з підтримкою багатьох портів введення-виведення та мультиплексування даних.

Наприклад, в роботі [23] запропоновано використання множини регістрів зсуву та відповідних мультиплексорів для розбиття вхідного потоку даних на паралельні шляхи та подальшу обробку в розподіленій пам'яті з підтримкою багатьох портів введення-виведення.

Цей підхід забезпечує паралелізм шляхів даних та підтримує класи алгоритмів, які вимагають доступу до різних регіонів.

Окрім цього, для покращення використання пам'яті можна використовувати архітектури пам'яті, що спеціалізуються на конкретних алгоритмах, наприклад, архітектури пам'яті, що використовуються в алгоритмах розпізнавання образів, можуть забезпечувати швидкий доступ до даних та зменшення кількості зайвих операцій.

Деякі роботи також пропонують використання додаткових апаратних блоків, таких як FPGA, для розподіленої обробки даних та зменшення використання ресурсів.

Загалом, використання оптимізованих архітектур пам'яті та підходів до розподіленої обробки даних може допомогти покращити продуктивність та енергоефективність систем обробки відео та зображень.

Однак, вибір підходу залежить від конкретної задачі та обмежень ресурсів, тому потрібно проводити детальний аналіз перед вибором оптимального рішення.

2.4 Системи кешування

Делегування зберігання кадрів позачиповій пам’яті є важливим аспектом при розробці пристроїв з високою продуктивністю та обмеженою ємністю вбудованої пам’яті.

На сьогоднішній день, збільшення ємності вбудованої пам'яті вимагає значних витрат на розробку та виготовлення.

Завдяки делегуванню зберігання кадрів позачиповій пам’яті, вирішується проблема ємності за рахунок продуктивності та грошових витрат.

Одним з методів кешування є використання паралельних відповідних масивів для прискорення обчислень.

Проте, кожен масив може утримувати лише один цікавий рядок, тому для зберігання повного кадру використовується зовнішня пам'ять.

Підхід з кешуванням може підтримувати лише обмежений клас алгоритмів, що ускладнює оцінку точної вартості масиву.

Наприклад, операції по стовпцях вимагають перевпорядкування позачіпової пам’яті, щоб дані завантажувалися на мікросхему у вигляді рядків, що споживає дорогоцінний час обробки.

Проте, є інші підходи до зберігання кадрів, які можуть бути значно ефективніші.

Наприклад, Chou et al. [23] показали використання векторної блокнотної

пам’яті для прискорення векторної обробки на FPGA, але все ще покладаються на зовнішню пам'ять з довільним доступом.

Подібного підходу дотримуються Naylor et al. [37] в контексті FPGA як прискорювачів.

Крім того, для вирішення проблеми ємності можуть бути використані інші методи, такі як компресія даних, що дозволяє зменшити об’єм необхідної пам’яті для зберігання кадрів.

Наприклад, можна використовувати алгоритми, які виявляють повторювані малі фрагменти у кадрах та зберігають їх лише один раз з подальшим використанням у всіх інших місцях.

Також можуть бути використані методи прогнозування, які дозволяють передбачити наступний кадр з високою точністю на основі попередніх кадрів та інших параметрів вхідних даних.

Отже, для вирішення проблеми ємності позачипової пам'яті можна використовувати різноманітні методи, такі як делегування зберігання кадрів в зовнішню пам'ять, компресія даних та прогнозування.

Кожен з цих методів має свої переваги та недоліки, тому вибір конкретного методу залежить від конкретної ситуації та потреб користувача.

Загалом, збільшення ємності позачипової пам'яті може покращити продуктивність та ефективність роботи системи, що має важливе значення для багатьох застосувань, включаючи обробку відеоданих та машинне навчання.

2.5 Алгоритми розбиття

Для проектів на основі HLS алгоритми комп’ютерного зору природно виражаються припущенням, що кадри зберігаються в необмежених адресних просторах.

Цей програмний підхід до проектування FPGA не тільки легко перевищує можливості пам’яті FPGA, але також нелегко інтегрується в проекти потокового передавання без значного рефакторингу.

Це призвело до розробки користувацьких апаратних блоків та API для інтеграції програмного забезпечення: «простий» HLS на основі C призводить до кількох структур пам’яті на чіпі, чиї розміри та інтерфейси залежать від типів змінних, часто з використанням наявних вбудована пам'ять.

Більшість інструментів HLS пропонують директиви компілятора — прагми, — які керують інструментом синтезу відповідно до задуму розробника: оптимізація продуктивності через розгортання циклу або вибір різних реалізацій (пам’яті на чіпові або LUT).

Відповідно, можна дійти висновку, що директиви, які використовують різні стратегії синтезу, більше потрібні для того, щоб впоратися з обмеженнями дизайну, такими як простір і потужність.

У зв'язку з тим, що HLS є одним із найважливіших програмних підходів для проектування FPGA, що має великий потенціал у різних сферах, наприклад, в медичних системах, автономних транспортних засобах та інших високотехнологічних пристроях, розробники стали звертати більше уваги на розвиток інструментів та методів для підтримки інтеграції і синтезу HLS в сучасних проект.

Однак, при використанні HLS є певні обмеження, пов'язані з якістю виводу.

Якщо вхідні дані містять помилки, то ці помилки можуть бути ускладнені під час генерації коду, що може призвести до погіршення якості результату. Крім того, HLS є досить новою технологією, тому для деяких задач може бути важко підібрати оптимальні параметри синтезу.

Також, слід відзначити, що при використанні HLS можуть виникати проблеми з надійністю та безпекою.

Якщо програмне забезпечення, що генерує код HLS, містить помилки або уразливості, то це може призвести до збоїв в роботі пристрою або навіть до його підриву.

Тому, як і в будь-якій іншій технології, важливо дотримуватися правил безпеки та надійності при використанні HLS.

У цілому, HLS є потужним інструментом для розробки апаратних пристроїв,

який дозволяє значно зменшити час розробки і полегшити роботу розробників, але при цьому вимагає деяких знань та навичок для ефективного використання.

Інші дослідження в галузі алгоритмів поділу зосереджуються на інших аспектах, таких як енергоефективність та зниження витрат на виробництво пам'яті.

Наприклад, в [17] розробили алгоритм поділу, який забезпечує зниження витрат на виробництво пам'яті шляхом мінімізації кількості використовуваних транзисторів. Автори також продемонстрували, що їхній алгоритм може знизити споживання енергії пам'яті до 54% порівняно з іншими алгоритмами поділу.

Окрім цього, деякі дослідження стосуються використання різних типів пам'яті для підвищення ефективності алгоритмів поділу.

Наприклад, Seem та ін. [18] використовували графенову пам'ять замість традиційної пам'яті на основі кремнію. Вони продемонстрували, що графенова пам'ять може забезпечити значні переваги в швидкості і енергоефективності.

Отже, в галузі алгоритмів поділу існує багато досліджень, які зосереджуються на різних аспектах покращення ефективності, енергоефективності та зниженні витрат на виробництво пам'яті.

Досягнення в цих галузях можуть привести до значного покращення продуктивності та ефективності систем, які використовують алгоритми поділу.

Тому, щоб забезпечити оптимальне використання пам'яті, необхідно проводити додаткові дослідження та розробляти нові методи. Один з можливих напрямів - це використання методів машинного навчання для автоматичного розподілу пам'яті в потокових програмах.

Наприклад, дослідження, проведені в [17], використовують глибинне навчання для автоматичного розподілу пам'яті на потокових архітектурах.

Вони показали, що їхній метод може підвищити ефективність використання пам'яті до 33,7% порівняно зі звичайним методом.

Інший можливий напрям - це використання технологій керування пам'яттю, таких як garbage collection, що дозволяють автоматично вивільняти незадіяну пам'ять та уникнути переповнення пам'яті.

Враховуючи все вищезазначене, можна зробити висновок, що оптимізація

використання пам'яті є важливим завданням для покращення продуктивності потокових програм.

Для досягнення цієї мети необхідно проводити дослідження та розробляти нові методи, такі як використання машинного навчання та технологій керування пам'яттю.

2.6 Висновки

Недавні дослідження показали, що використання кеш-пам'яті зменшує енергоспоживання за рахунок зменшення затримок при доступі до пам'яті.

Дана робота досліджує вплив розподілу пам'яті на обмежену доступність у мобільних пристроях та вбудованих системах.

Для цього розроблено алгоритми розділення, які оптимізують використання пам'яті з урахуванням обмежень на доступність.

Було визначено розміщення даних та здійснено розподіл пам'яті таким чином, щоб мінімізувати кількість затримок при доступі до пам'яті та забезпечити максимальну продуктивність.

В результаті алгоритми розділення дозволяють знизити енергоспоживання та збільшити продуктивність вбудованих систем, які обробляють вхідні зображення.

Для валідації алгоритмів проведено експерименти на реальних пристроях та порівняно їх продуктивність з існуючими рішеннями.

Запропонований метод побудови архітектури вбудованих систем обробки зображень на базі FPGA є ефективним і може бути використаний для створення потужних вбудованих систем обробки зображень з мінімальним споживанням енергії та високою продуктивністю.

Дослідження має великий потенціал для використання в різних областях, що потребують обробки зображень з високою швидкодією та низьким споживанням енергії.

В даному розділі були досліджені методи побудови архітектури вбудованих систем обробки зображень на основі FPGA та проаналізовані сучасні програмно-

технічні засоби для обробки зображень на основі FPGA.

Досліджені компоненти стануть основою методу побудови архітектури вбудованих систем обробки зображень на основі FPGA.

3 МЕТОД ПОБУДОВИ АРХІТЕКТУРИ ВБУДОВАНИХ СИСТЕМ ОБРОБКИ ЗОБРАЖЕНЬ НА ОСНОВІ FPGA

3.1 Ефективність використання

Для вирішення поставленої мети — метод було вдосконалено за рахунок побудови архітектури вбудованих систем обробки зображень, який на відміну від відомих використовує FPGA та грунтується на мінімізації потужності, та оптимізованому використанні пам’яті, і який забезпечує високу швидкодію обробки зображень.

Враховуючи ємність зберігання BRAM C і кількість можливих конфігурацій «i», набір конфігурацій Cfg є вектором «i» елементів:

$$Cfg = \begin{pmatrix} (M_1, N_1) \\ (M_2, N_2) \\ . \\ . \\ . \\ (M_i, N_i) \\ \end{pmatrix} = \begin{pmatrix} Cfg_1 \\ Cfg_2 \\ . \\ . \\ . \\ Cfg_i \\ \end{pmatrix} \tag{3.1}$$

Де перший компонент кожного елемента відображає ширину BRAM M, а другий компонент зображує висоту BRAM N, так що:

$$M_x * N_x \leq C, \forall_x \in [0, i-1]. \tag{3.2}$$

Для будь-якого даного розміру кадру можливі кілька можливих топологій BRAM (різні конфігурації BRAM не завжди відповідають однаковій логічній ємності.

Хоча загальна фізична ємність однакова, у деяких конфігураціях біти

парності можуть використовуватися як додаткові біти даних. Наприклад, конфігурація (1,16384) може зберігати 16384 біти, тоді як конфігурація (9,2048) може зберігати 18432 біти). Кадр — це тривимірний масив із шириною W, висотою H і піксельною бітовою шириною Bw (зазвичай визначається як 2-вимірний масив, де тип визначає розмірність бітової ширини).

Топології BRAM визначаються на основі відображення 3-D у 2-D масивів і розбиття 2-D масиву на певну структуру пам’яті, див. рисунок 3.1.

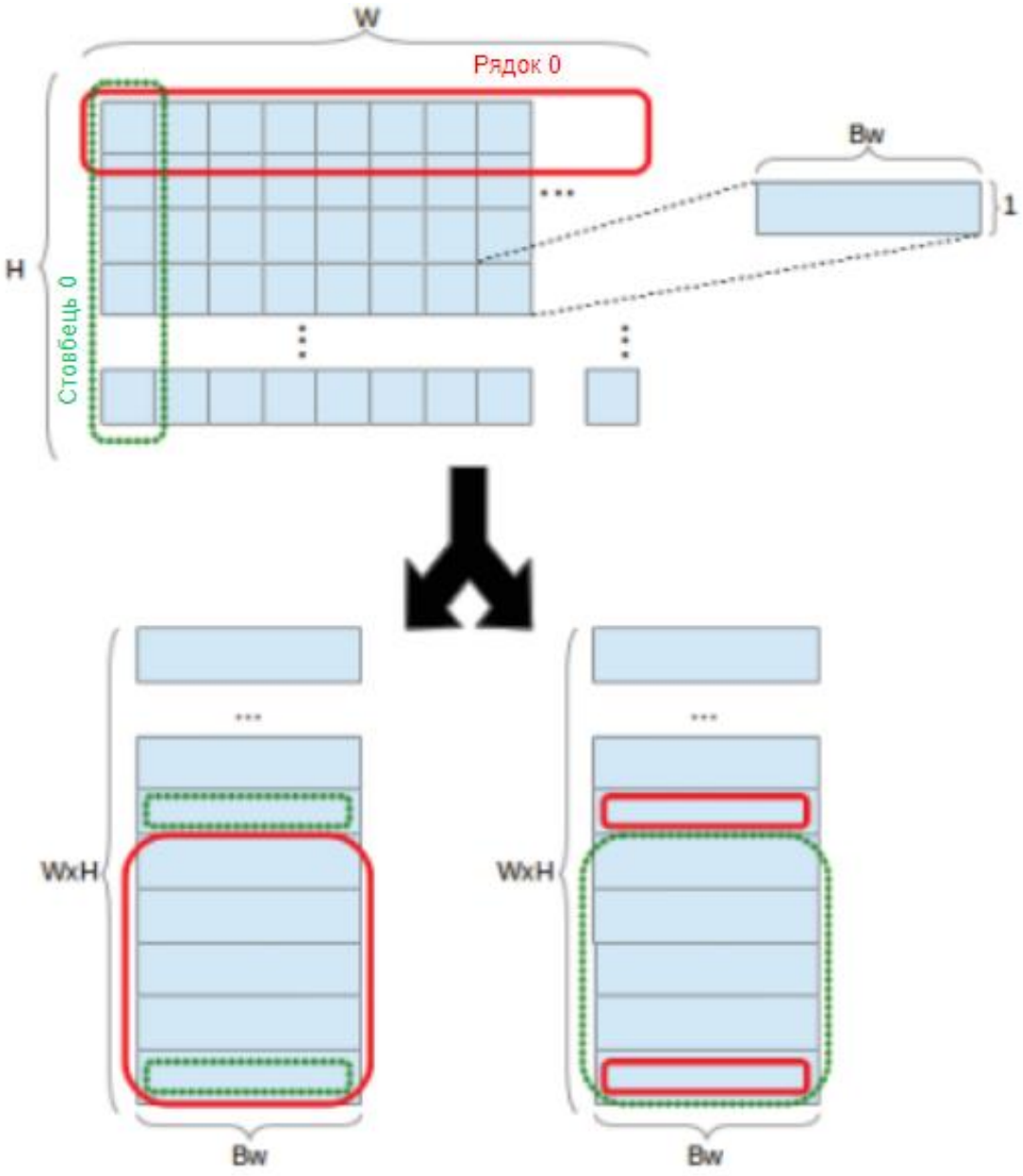


Рисунок 3.1 - Відображення тривимірного масиву в двовимірні масиви в порядку основних рядків і стовпців.

Можна припустити використання схеми відображення, яка призначає Bw виміру x, а H і W – виміру y, як у порядку рядків, так і в порядку стовпців (де x і y є 2-D ширина та висота масиву відповідно).

Це підхід за замовчуванням у реалізаціях програмного забезпечення, де розмір типу/розрядності вважається неявним, і розумний підхід для реалізації апаратного забезпечення.

Відображення бітової ширини Bw через розмір y призведе до реалізацій, де різні біти одного елемента масиву (пікселя) будуть розкидані між різними позиціями пам’яті однієї BRAM.

Для цього знадобиться послідовна логіка для читання/запису пікселя, доступу до кількох позицій пам’яті, створення накладних витрат на продуктивність, потужність і розмір.

Слід зазначити, що цей підхід може запропонувати переваги продуктивності для певних класів алгоритмів, які можуть захотіти порівнювати окремі біти різних елементів. Потрібно, визначити схему відображення за замовчуванням:

Схема відображення m перетворює тривимірний масив A3 у двовимірний масив A2 розмірів x і y шляхом призначення Bw розміру x і впорядкованих комбінацій W і H розміру y, для всього двох можливих конфігурацій, як показано на рисунку 3.1. Схеми відображення визначаються як:

$$(x, y) = m(W, H, B_\omega).$$
$$A2_{x,y} = A3_{y\backslash W, y\%W, x'}\ x = B_{\omega'} y = W * H.$$
$$A2_{x,y} = A3_{y\%H, y\backslash H, x'}\ x = B_{\omega'} y = W * H. \quad (3.3)$$

Де \ та % представляють ціле числове ділення та модуль відповідно.

Враховуючи двовимірний відображений кадр зображення з розмірами x і y, схема поділу p, яка призначає пікселі через a × b BRAM, зображена на рисунку 3.2, визначається як лінійна комбінація:

$$p(x,y) = Cfg * \big((a_1,b_1),(a_2,b_2),\dots,(a_i,b_i)\big).$$

(3.4)

Де $*$ позначає лінійну комбінацію, таку, що лише одна пара (ax, bx), $\forall x \in [0, i-1]$ має ненульові компоненти (така пара генерується як функція x та y), вибираючи Mp і Np з урахуванням:

$$((a * M_p) \geq x) \cap ((b * N_p) \geq y).$$

(3.5)

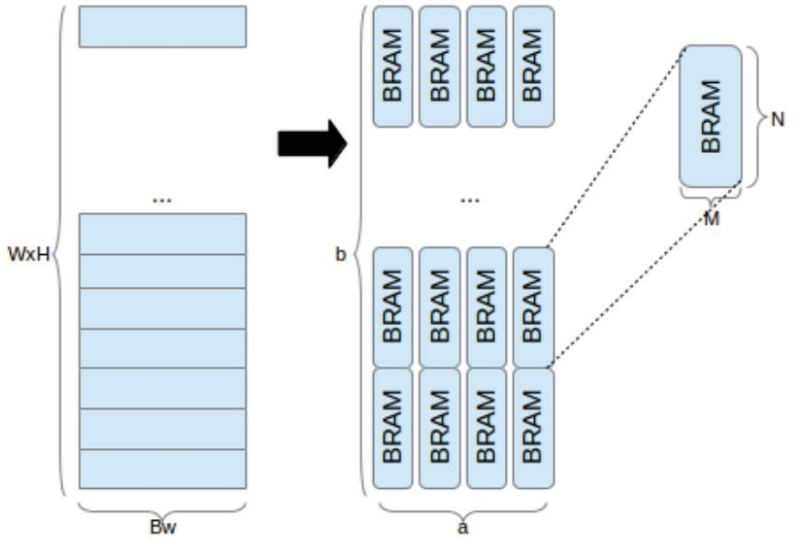


Рисунок 3.2 - Відображення двовимірного масиву розмірів x = Bw і y = W × H на BRAM a × b, налаштованих на ширину M і висоту N.

Різні схеми поділу p, що реалізують різні функції x і y, призводять до різних вимог до адресації, логіки введення та виведення, кожна з яких має особливий вплив на продуктивність і використання ресурсів.

Оскільки це найбільше вузьке місце у впровадженні високорівневих конвеєрів обробки зображень на FPGA, надзвичайно важливо визначити ефективність використання BRAM, тобто співвідношення між загальною ємністю даних призначених BRAM та кількістю даних, які фактично використовуються.

Враховуючи схему розподілу p і максимальну потужність BRAM C, ефективність використання E визначається як співвідношення:

$$E = \frac{x * y}{a_p * b_p * C} \tag{3.6}$$

Схеми відображення та розділення за замовчуванням у найсучасніших інструментах HLS спрямовані на мінімізацію логіки адресації (багато в сучасних FPGA), що призводить до низької ефективності використання BRAM (все ще мало для вимог систем обробки зображень високого рівня). Необхідно використовувати альтернативні схеми, щоб забезпечити доступність пам'яті в рамках проектних потоків HLS. Ми визначаємо проблему як:Враховуючи рамку зображення шириною W, висотою H і шириною пікселів Bw, вибираємо схему поділу, щоб:

$$Maximize\ E = \frac{x * y}{a_p * b_p * C}$$
$$Subject\ to\ ((a * M_p) \geq x) \cap ((b * N_p) \geq y). \tag{3.7}$$

Розглядається кадр зображення шириною W = 320 і висотою H = 240, де кожен піксель має 8 біт (монохромний), і BRAM, які можна налаштувати відповідно до:

$$Cfg = \begin{pmatrix} (1{,}16384) \\ (2{,}8192) \\ (4{,}4096) \\ (9{,}2048) \\ (18{,}1024) \\ (36{,}512) \end{pmatrix} \tag{3.8}$$

Який є репрезентативним для найсучасніших FPGA (Altera Cyclone V сімейства 18Kbits BRAM.), де загальна ємність BRAM C визначається як C = 36 × 512. Використання схеми розділення.

$$p\big(m(320{,}240{,}8)\big) = Cfg * \begin{pmatrix} \\ (8{,}8) \\ (0{,}0) \\ (0{,}0) \\ (0{,}0) \\ (0{,}0) \\ (0{,}0) \end{pmatrix}^{T} \tag{3.9}$$

Де m(320, 240, 8)=(8, 76800) (рівняння (3)), дає кількість використання BRAM 64 (8 × 8 BRAM, налаштованих на ширину 1 і висоту 16384), з ефективністю зберігання:

$$E = \frac{8*(320*240)}{8*8*(36*512)} = 0.520833333. \tag{3.10}$$

Це поведінка за замовчуванням для інструментів синтезу Altera Cyclone HLS: емпіричні результати показують, що конфігурація (M1, N1)=(1, 16384) вибирається через схему розділення, де a1 = Bw

$$b_1 = \frac{W * H}{N_1} \tag{3.11}$$

Округлені до найближчого ступеня 2.

Експерименти показують, що для будь-якого розміру кадру типову схему розділення інструментів синтезу можна задати так:

$$p\big(m(W,H,B_\omega)\big) = Cfg * \begin{pmatrix} (B_\omega, 2^{[log_2(\frac{W*H}{N_1})]}) \\ (0,0) \\ (0,0) \\ (0,0) \\ (0,0) \\ (0,0) \end{pmatrix}^T \tag{3.12}$$

Де 2 log2(W × H N1) слід читати як 2 до округленого (максимального) результату операції логарифмування (тобто 2 до цілого степеня).

Потрібно розглянути те саме відображення (x = Bw, y = W × H), але зі схемою розбиття:

$$P\big(m(320,240,8)\big) = Cfg * \begin{pmatrix} (8,5) \\ (0,0) \\ (0,0) \\ (0,0) \\ (0,0) \\ (0,0) \end{pmatrix}^T \tag{3.13}$$

Яка розділяє дані нерівномірно між BRAM, а не рівномірно. Ця схема дає кількість використання BRAM 40 із ефективністю зберігання:

$$E = \frac{320 * 240 * 8}{8 * 5 * (36 * 512)} = 0.833333333. \tag{3.14}$$

Проте кращою схемою поділу для того самого відображення буде:

$$P\big(m(320,240,8)\big) = Cfg * \begin{pmatrix} \\ (0,0) \\ (0,0) \\ (2,19) \\ (0,0) \\ (0,0) \\ (0,0) \end{pmatrix}^{T} \tag{3.15}$$

Показник BRAM 38 і ефективність:

$$E = \frac{320 * 240 * 8}{2 * 19 * (36 * 512)} = 0.977192982. \tag{3.16}$$

Зрозуміло, що схеми розділення залежать від розмірів кадру, ширини, висоти та розрядності, щоб забезпечити ефективне використання блоків пам’яті на чіпові.

3.2 Дослідження значень живлення

Формалізувавши проблему використання, можна приступити до аналізу енергетичних наслідків кожної конфігурації.

Потрібно змоделювати динамічне енергоспоживання BRAM. Квант потужності споживається на читання та/або запис. Статична потужність BRAM прямо пропорційна використанню, тому розглядається в проблемі використання.

Для будь-якої комірки BRAM потужність зчитування споживається послідовністю операцій: тактовий сигнал стробується; прочитана адреса декодується; зчитані дані стробуються в колонковий мультиплексор; зчитані дані передаються на зовнішній порт BRAM.

Потужність запису споживається в такій послідовності: тактовий сигнал стробується; сигнал дозволу запису передає дані запису в буфери запису; рядок виділяється розшифровкою адреси; дані зберігаються в комірці RAM.

Далі розглядається поділ, де кожна дана розподілена між вісьмома BRAM, і поділ, де кожна дана розподілена між двома BRAM.

Кожна операція читання/запису в першому повинна споживати енергію в чотири рази більше кількості BRAM в останньому. На рисунках 3.3 і 3.4 зображено приклади споживання електроенергії для двох схем розподілу.

Схема поділу, яка мінімізує горизонтальне використання BRAM (тобто через x), більше підходить для стробування тактового сигналу.

Оскільки за одну операцію необхідно отримати доступ до меншої кількості блоків BRAM, пропорція невикористаних блоків, які можуть бути ефективно закриті, збільшується.

Це просто реалізувати тактовий строб за допомогою вибору дозволу мікросхеми, який увімкнено/вимкнено на основі декодування адреси. colorred Іншими словами, енергоспоживання BRAM пропорційне кількості BRAM, необхідних для доступу до кожного пікселя: і це число залежить від обраної конфігурації.

Інтуїтивно зрозумілий підхід до балансування енергоспоживання та використання полягає в тому, щоб завжди використовувати найширшу конфігурацію BRAM, достатньої для Bw, або кратні найширшій доступній.

Однак це не оптимізована стратегія. Хоча це правда, що динамічна потужність зменшується, статична потужність може збільшуватися при переході від однієї конфігурації до ширшої, оскільки загальна кількість BRAM може збільшитися: ефективність використання змінюється.

Крім того, логіка, необхідна для сигналів адреси (і мікросхеми), збільшується при переході до ширшої конфігурації. Цей аспект робить проблеми використання та живлення неподільними. Таким чином необхідне балансування цих двох аспектів.

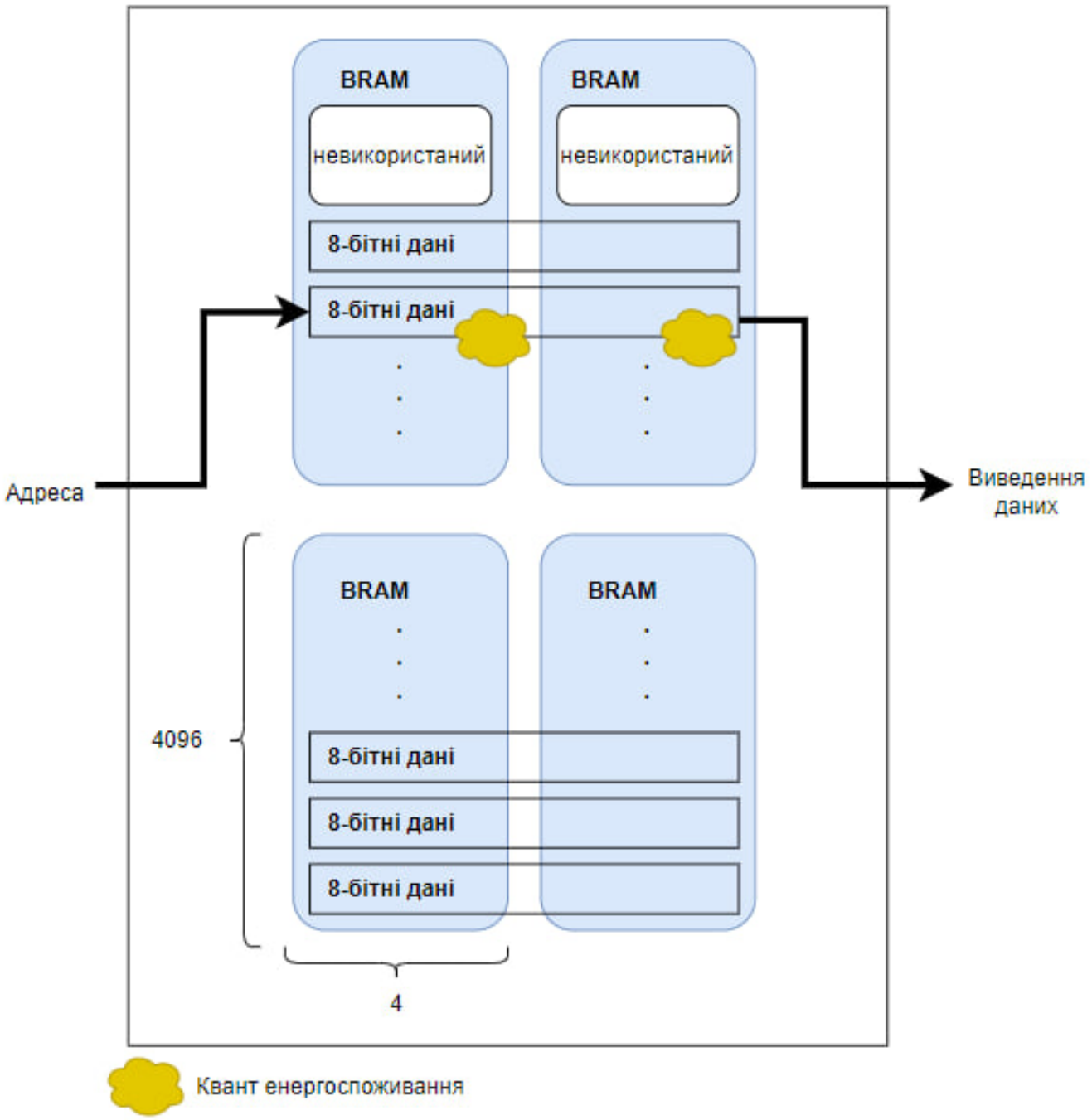


Рисунок 3.3 - Розбиття на дві горизонтальні частини між двома блоками запам'ятовування з випадковим доступом (BRAM). Кожен доступ використовує дві одиниці кількості споживання енергії

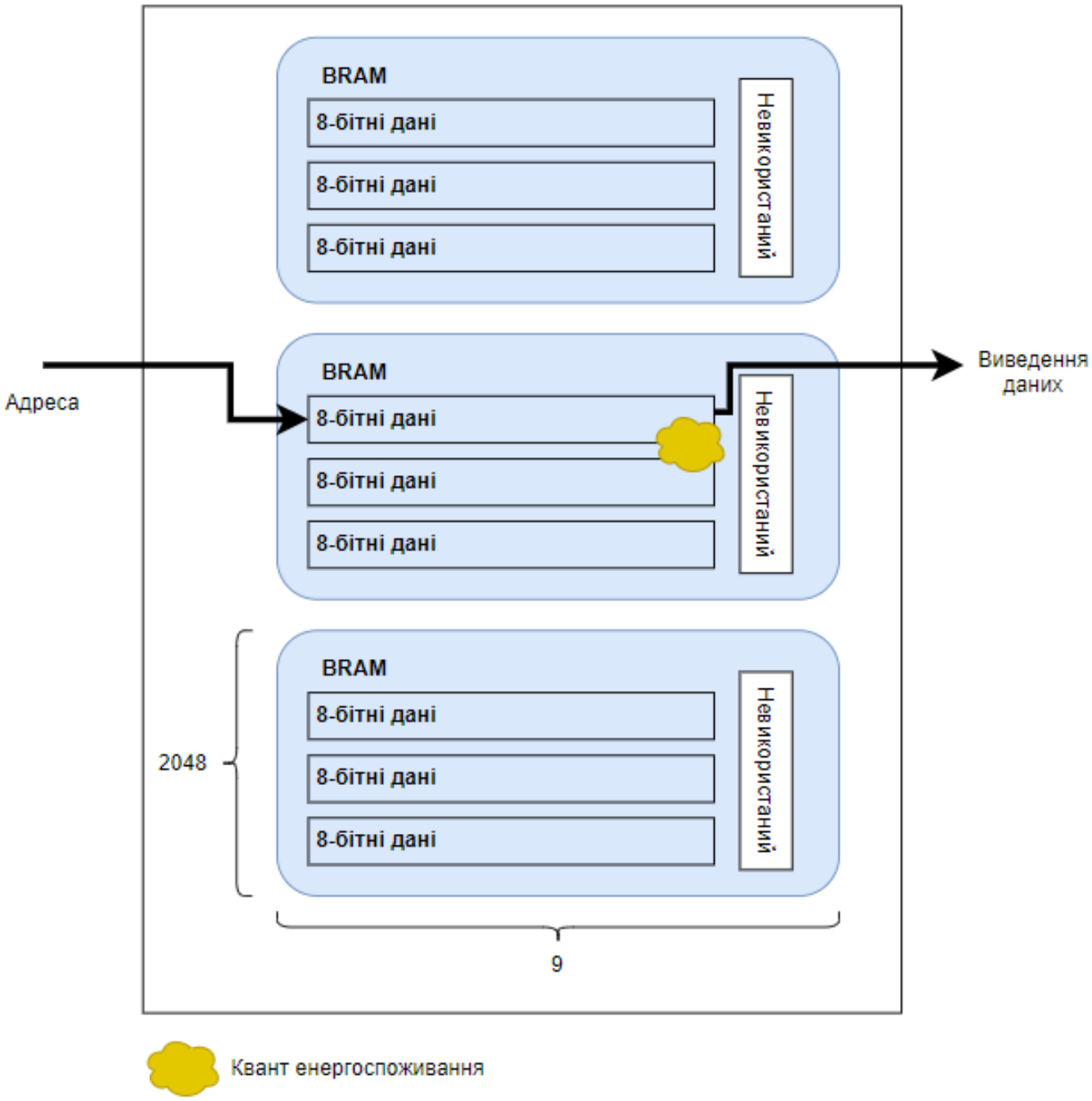


Рисунок 3.4 - Розбиття на горизонтальні частини в межах одного блоку запам'ятовування з випадковим доступом (BRAM). Кожен доступ використовує одну одиницю кількості споживання енергії

3.3 Розбиття для живлення та використання

Для вирішення задачі підвищення швидкодії систем обробки зображень на

основі FPGA необхідно приступити до аналізу енергоспоживання кожної конфігурації.

Варто почати з представлення оптимізованої грубою силою процедури розділення для максимізації ефективності використання, описаного нижче псевдокоду.

procedure OPTIMIZED PARTITION
efficiency ← 0
best ← 0
for x=0 : i-1 do
(Mx,Nx) ← Cfgx
a ← Bw/Mx
b ← W × H/Nx
efficiency ← (W × H × Bw)/(a × b × C)
if efficiency greater than best then
best ← ef ficiency
configuration ← (Mx, Nx)
end if
end for

Для кожного елемента в наборі конфігурацій Cfg (що має загалом і елементів) процедура обчислює необхідну кількість BRAM для зберігання кадру шириною W, висотою H і шириною Bw, ефективність такої конфігурації та порівнює її з найвища ефективність, знайдена на даний момент.

Тут увага зосереджена виключно на використанні. По суті, це вичерпний пошук, оскільки кількість можливих конфігурацій пам’яті обмежена, і це автономний процес.

У таблиці 3.1 зображено конфігурації, вибрані процедурою для репрезентативної кількості розмірів кадру та бітової ширини пікселів.

Деякі з конфігурацій не оптимізовані за енергоспоживанням: зауважте, що

для пікселів шириною 10, 14 і 22 найчастіше вибирається конфігурація BRAM 2 × 8192 (споживає потужність на 5, 7 і 11 BRAM на доступ відповідно). Це інтуїтивно зрозуміло з точки зору ефективності використання: це єдина конфігурація, яка розділяє ширину, і відповідає вибору конфігурації 4 × 4096 для пікселів шириною 8, 12, 20 і 24 і конфігурації 18 × 1024 для пікселів ширини. ширина 18.

Таблиця 3.1 - Конфігурації BRAM на основі оптимізованої процедури використання.

Ширина пікселів									
Кадр	8	10	12	14	16	18	20	22	24
160 x 120	4 x 4096	4 x 4096	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	4 x 4096
320 x 240	4 x 4096	2 x 8192	4 x 4096	2 x 8192	18 x 1024	18 x 1024	4 x 4096	2 x 8192	4 x 4096
512 x 512	4 x 4096	2 x 8192	4 x 4096	2 x 8192	4 x 4096	18 x 1024	4 x 4096	2 x 8192	1 x 16384
640 x 480	4 x 4096	2 x 8192	4 x 4096	2 x 8192	18 x 1024	18 x 1024	4 x 4096	2 x 8192	4 x 4096
1280 x 720	4 x 4096	2 x 8192	4 x 4096	2 x 8192	18 x 1024	18 x 1024	4 x 4096	2 x 8192	4 x 4096

Ця нелінійність ускладнює розробку оптимізованої процедури розподілу як за використанням, так і за енергоефективністю.

Таким чином, можна використовувати більш спокійний підхід і визначити процедуру через визначені користувачем компроміси (тобто оцінку того, скільки використання BRAM можна обміняти на зменшення потужності) і евристику потужності та простору на основі емпіричних властивостей. Передбачається, що компроміс виражається у відсоткових пунктах.

Процедура починається з вибору оптимізованого рішення щодо використання та повторення ширших конфігурацій BRAM (у вимірі x), обчислення ефективності використання.

Поки рівень використання перевищує порогове значення, визначене різницею між найкращим використанням і компромісом у відсоткових пунктах, процедура продовжується. Коли він знаходить перше рішення нижче порогового значення, він виходить, повертаючи останнє рішення вище порогового значення.

Цей підхід відповідає евристиці моделі потужності, описаній у раніше: енергоспоживання зменшується зі збільшенням горизонтальної ширини BRAM, див. рис. 3.3 і 3.4. Збалансований метод грубої сили описано в наступному алгоритмі.

procedure BALANCED PARTITION
efficiency ← 0
configuration ← get_MxNx(OptimizedPartition())
best ← get_efficiency(OptimizedPartition())
j ← get_index(OptimizedPartition())
for x=j+1 : i-1 do efficiency
(Mx,Nx) ← Cfgx
a ← Bw/Mx
b ← W × H/Nx
efficiency ← (W × H × Bw)/(a × b × C)
if efficiency less than best - tradeoff then
break
end if
configuration ← (Mx, Nx)
end for

У таблиці 3.2 зображено конфігурації BRAM, вибрані за збалансованою процедурою, із встановленим компромісом у 12 відсоткових пунктів. Порівняно з оптимізованими конфігураціями, більшість ширини збільшено, що призводить до більш енергоефективного рішення на основі вищезгаданої евристики.

3.4 Застосування розділення пам'яті

Вищеописані алгоритми можна використовувати як у потоках проектування HDL, так і в потоках проектування HLS: у потоці проектування HDL, керуючи впровадженням та/або рефакторингом дизайнера; у потоці проектування HLS через інтеграцію в підсистему генерації коду інструментів синтезу.

Таблиця 3.2 - Конфігурації BRAM на основі збалансованої процедури з компромісом, рівним дванадцяти відсотковим пунктам

Ширина пікселів									
Кадр	8	10	12	14	16	18	20	22	24
160 x 120	9 x 2048	4 x 4096	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048
320 x 240	9 x 2048	4 x 4096	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048
512 x 512	9 x 2048	2 x 8192	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048
640 x 480	9 x 2048	2 x 8192	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048
1280 x 720	9 x 2048	2 x 8192	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048

На рисунку 3.5 зображено запропоновані проектні потоки. Додаткові кроки можна виконувати вручну, або починаючи з проектів HDL, або змінюючи вихідні дані HLS перед синтезом; за допомогою автоматизованих засобів рефакторингу, які обчислюють запропоновані процедури; або інструментом HLS до генерації коду. Описується ручний процес, який використовується в наших експериментах.Після того, як структуру пам’яті було отримано зі специфікації процедури, відповідно до рівнянь та алгоритмів, обчислюються для визначення розподілу BRAM. BRAM обчисленої конфігурації створюється та міститься в модулях (тобто апаратних

об’єктах). Верхній модуль створює екземпляри всіх підмодулів, надаючи інтерфейси, ідентичні базовому дизайну HDL або специфікації інструменту HLS. Логіка адресації всередині верхнього модуля керує мікросхемою вмикання сигналів для кожного підмодуля, гарантуючи, що неадресовані BRAM не ввімкнено. Цей ретельний розподіл логіки HDL в ієрархічних модулях, де логіка адресації визначається міжз’єднанням верхнього рівня, а конфігурація BRAM визначається параметрами конфігурації модуля, забезпечує використання бажаних конфігурацій (це базується на наших експериментах із використанням Vivado: різні інструменти синтезу можуть вимагають додаткових прагм компілятора).

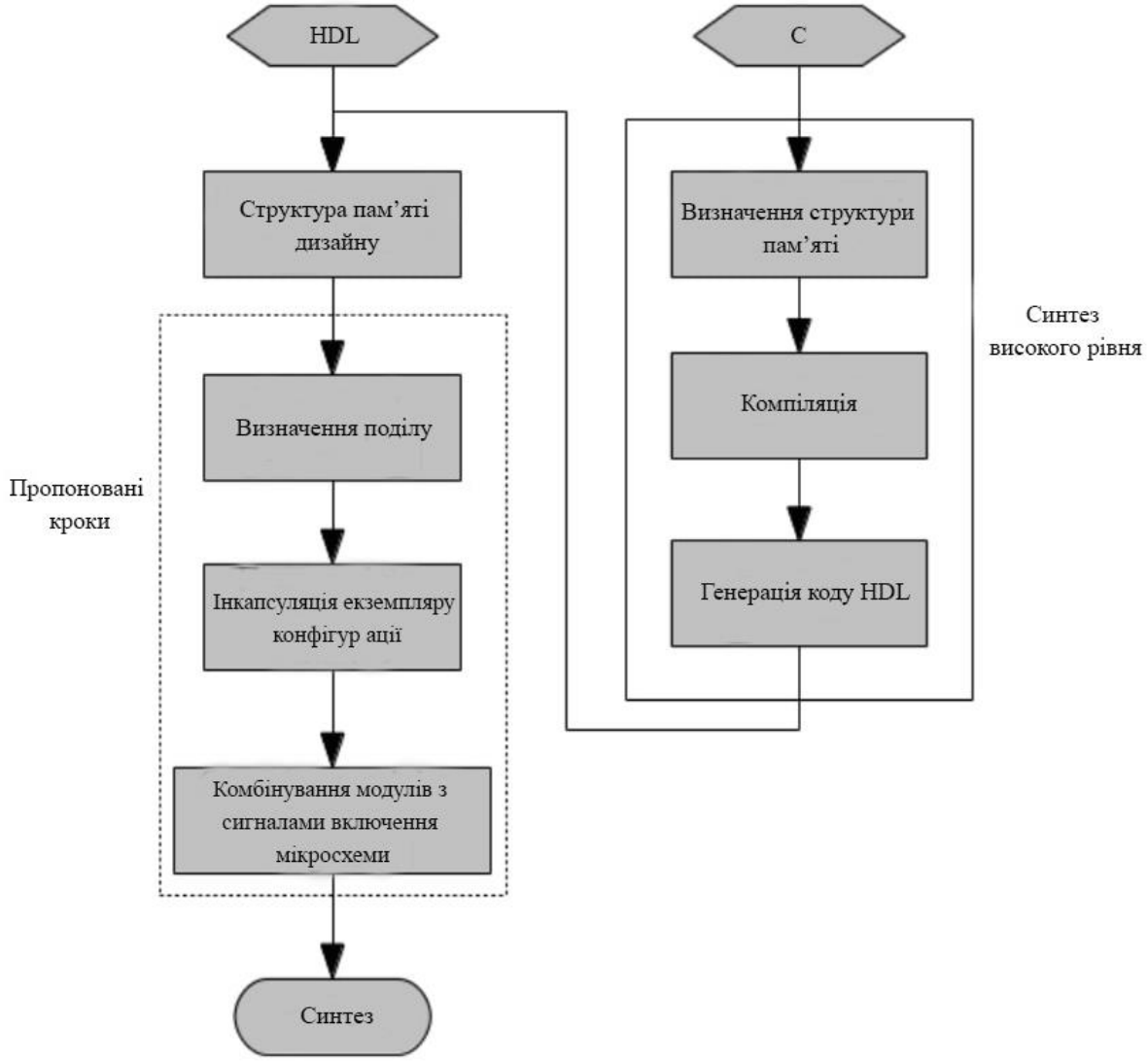


Рисунок 3.5 - Запропонований проектний потік від HDL і HLS, підкреслюючи додаткові кроки, необхідні для мінімізації використання та потужності

3.5 Висновки

У цьому розділі було представлено метод побудови архітектури вбудованих систем обробки зображень на основі FPGA. Наукова новизна цього методу полягає в удосконаленні методу побудови архітектури вбудованих систем обробки зображень, який на відміну від відомих використовує FPGA та грунтується на мінімізації потужності, та оптимізованому використанні пам’яті, і який забезпечує високу швидкодію обробки зображень.

Був проведений опис розділення кадрів зображень на BRAM, щоб максимізувати використання (тобто мінімізувати кількість необхідної пам’яті на чіпі) за умови мінімізації споживання енергії. Було зформульовано проблему ефективності використання, не звертаючи жодної уваги на аспекти потужності. Також було інтегровано енергоспоживання в формулювання проблеми.

В результаті було дійдено до висновку, що для кожного кадрового буфера зображення використовується лише одна можлива конфігурація BRAM.

В даному розділі було розроблено модель функціонування програмно-технічних засобів обробки зображень на основі FPGA та розроблено метод побудови архітектури вбудованих систем обробки зображень на основі FPGA;

4 РЕАЛІЗАЦІЯ МЕТОДУ ПОБУДОВИ АРХІТЕКТУРИ ВБУДОВАНИХ СИСТЕМ ОБРОБКИ ЗОБРАЖЕНЬ НА ОСНОВІ FPGA

4.1 Буфери кадрів: вплив конфігурації BRAM

Дана робота спрямована на сучасні пристрої FPGA (пристрій Altera Cyclone V). В роботі використовується Vivado v2016.1 для проектування HDL, Vivado HLS v2016.1 для високорівневого синтезу та Early Power Estimator для характеристики потужності реалізованих проектів. Все починається з генерації буферів кадрів у кількох конфігураціях, щоб охарактеризувати ефективність використання та енергоспоживання. Потім порівнюється використання та потужність з еквівалентними буферами кадрів, створеними інструментом HLS. На завершення реалізовуються два високорівневі алгоритми обробки зображень через HLS і змінюються буфери кадрів відповідно до запропонованих стратегій, щоб кількісно визначити вплив алгоритмів на використання ресурсів і енергоспоживання в повних системах обробки зображень.

Перша частина дослідів характеризує використання та енергоспоживання для двох типорозмірів рами як функцію кількох можливих конфігурацій. Метою цього набору дослідів було перевірити ефективність використання алгоритмів розподілу та евристики потужності, використаних у попередньому розділі.

Було здійснено реалізацію буферів кадрів у Verilog HDL у Vivado v2016.1, явно створюючи екземпляри BRAM відповідно до бажаних конфігурацій. Логіка в ієрархії проектування відповідно направляє дані, адреси та сигнали керування. Аналіз звітів після впровадження було виконано для того, щоб переконатися, що BRAM були створені відповідно до бажаної конфігурації (залежно від ієрархії дизайну, оптимізація інструменту синтезу могла реально реорганізувати розподіл BRAM). Було проведено дослід із послідовним читанням/записом, у якому повний кадр записується в пам’ять (послідовне введення пікселів у порядку старшого рядка), а потім зчитується в тому ж порядку. Це дозволяє перевірити евристику моделі потужності, прийняту в попередньому розділі. У таблицях 4.1 та 4.2 наведено результати потужності та використання монохромних рамок розміром

320 × 240 і 512 × 512.

Таблиця 4.2 - Енергоспоживання FPGA та ефективність використання (Eff.) для монохроматичних (8 біт) кадрів розміром 320 × 240 для різних конфігурацій BRAM

320 x 240				
Конфігурація	Статика	Потужність (Вт) Динаміка	BRAM	Ефективність (%)
8 x 5-1 x 16384	0.328	0.054	0.036	83.33
4 x 10-2 x 8192	0.327	0.036	0.018	83.33
2 x 19-4 x 4096	0.327	0.026	0.009	87.72
1 x 38-9 x 2048	0.327	0.027	0.005	87.72

Таблиця 4.2 - Енергоспоживання FPGA та ефективність використання (Eff.) для монохроматичних (8 біт) кадрів розміром 512 × 512 для різних конфігурацій BRAM

512 x 512				
Конфігурація	Статика	Потужність (Вт) Динаміка	BRAM	Ефективність (%)
8 x 5-1 x 16384	0.332	0.07	0.036	88.88
4 x 10-2 x 8192	0.331	0.053	0.018	88.88
2 x 19-4 x 4096	0.331	0.043	0.009	88.88
1 x 38-9 x 2048	0.331	0.046	0.005	88.88

4.2 Буфери кадрів: порівняння HLS

Друга частина дослідів порівнює запропоновані алгоритми розподілу зі стратегіями за замовчуванням, що використовуються комерційними інструментами HLS.

Мета цього набору дослідів полягала в тому, щоб підтвердити, що запропонована методологія перевершує комерційні інструменти HLS як за

використанням, так і за енергоспоживанням.

Було виконано високорівневий синтез на основі C з використанням Altera Cyclone HLS, описуючи кадри в стандартному форматі (тип масиву визначає бітову ширину, індекси визначають ширину та висоту кадру). Для кожного розміру кадру повідомляється про використання BRAM і додаткові ресурси (реєстри фрагментів і LUT). В роботі використовується стандартна ширина пікселів (8 біт для монохромних зображень, 24 біти для RGB). Було проведено оцінку оптимізованого використання BRAM за допомогою оптимізованого алгоритму використання та відповідно до алгоритму збалансованого розподілу, щоб порівняти вплив потужності та використання — алгоритми запускалися в автономному режимі.

Буфери кадрів реалізовано у Verilog HDL відповідно до кожного алгоритму, гарантуючи, що зовнішні інтерфейси (тобто дані для читання/запису, адреси та порти сигналів керування) ідентичні тим, які генерує Vivado HLS із C. Далі було замінено буфери кадрів, згенеровані з HLS з ручними версіями Verilog HDL. Для кожного розміру кадру повідомляється про використання BRAM і додаткові ресурси (реєстри фрагментів і LUT), необхідні для реалізації логіки адресації.

У таблицях 4.2, 4.3 наведено результати, отримані за трьома конфігураціями для монохроматичних і RGB-кадрів відповідно, а на рисунку 4.1 порівнюється ефективність використання BRAM.

Наслідки споживання електроенергії охарактеризовані кожною згенерованою системою за допомогою Altera Power Estimator для шаблонів доступу, характерних для програм обробки зображень. У досліді з послідовним читанням/записом повний кадр записується в пам’ять (послідовне введення пікселів у порядку старшого рядка), а потім зчитується в тому ж порядку. У досліді з ковзним вікном повний кадр читається через ковзне вікно 3 × 3. На рисунку 4.2 зображено статичне енергоспоживання;

На рисунках 4.3 і 4.4 зображено загальне динамічне енергоспоживання трьома архітектурами для тестових випадків послідовного читання/запису та ковзного вікна відповідно; і рисунки 4.5 і 4.6 зображують енергоспоживання BRAM для тестових випадків послідовного читання/запису та ковзного вікна.

Таблиця 4.2 – Оптимізоване використання ресурсів FPGA для монохроматичних кадрів

8 біт	HLS		Оптимізоване використання			
Кадр	BRAMs	LUTs	BRAMs			LUTs
			Використання	Режим	Зниження	
160 x 120	16	0	10	4 x 4096	-37.5%	22
320 x 240	64	9	38	4 x 4096	-40.6%	79
512 x 512	128	17	128	4 x 4096	0%	285
640 x 480	256	34	150	4 x 4096	-41.4%	337
1280 x 720	512	64	450	4 x 4096	-12.1%	1039
24 біти	HLS		Оптимізоване використання			
Кадр	BRAMs	LUTs	BRAMs			LUTs
			Використання	Режим	Зниження	
160 x 120	48	0	30	4 x 4096	-37.5%	41
320 x 240	192	25	114	4 x 4096	-40.6%	140
512 x 512	384	49	384	1 x 16384	0%	504
640 x 480	768	98	450	4 x 4096	-41.4%	584
1280 x 720	1536	192	1350	4 x 4096	-12.1%	1760

Таблиця 4.3 – Оптимізоване використання ресурсів FPGA для монохроматичних кадрів

8 біт	HLS		Збалансоване використання			
Кадр	BRAMs	LUTs	BRAMs			LUTs
			Використання	Режим	Зниження	
160 x 120	16	0	10	9 x 2048	-37.5%	48
320 x 240	64	9	38	9 x 2048	-40.6%	186
512 x 512	128	17	128	9 x 2048	0%	596
640 x 480	256	34	150	9 x 2048	-41.4%	742
1280 x 720	512	64	450	9 x 2048	-12.1%	2284
24 біти	HLS		Збалансоване використання			
Кадр	BRAMs	LUTs	BRAMs			LUTs
			Використання	Режим	Зниження	
160 x 120	48	0	30	9 x 2048	-37.5%	91
320 x 240	192	25	114	9 x 2048	-40.6%	308
512 x 512	384	49	384	9 x 2048	0%	1109
640 x 480	768	98	450	9 x 2048	-41.4%	1285
1280 x 720	1536	192	1350	9 x 2048	-12.1%	3877

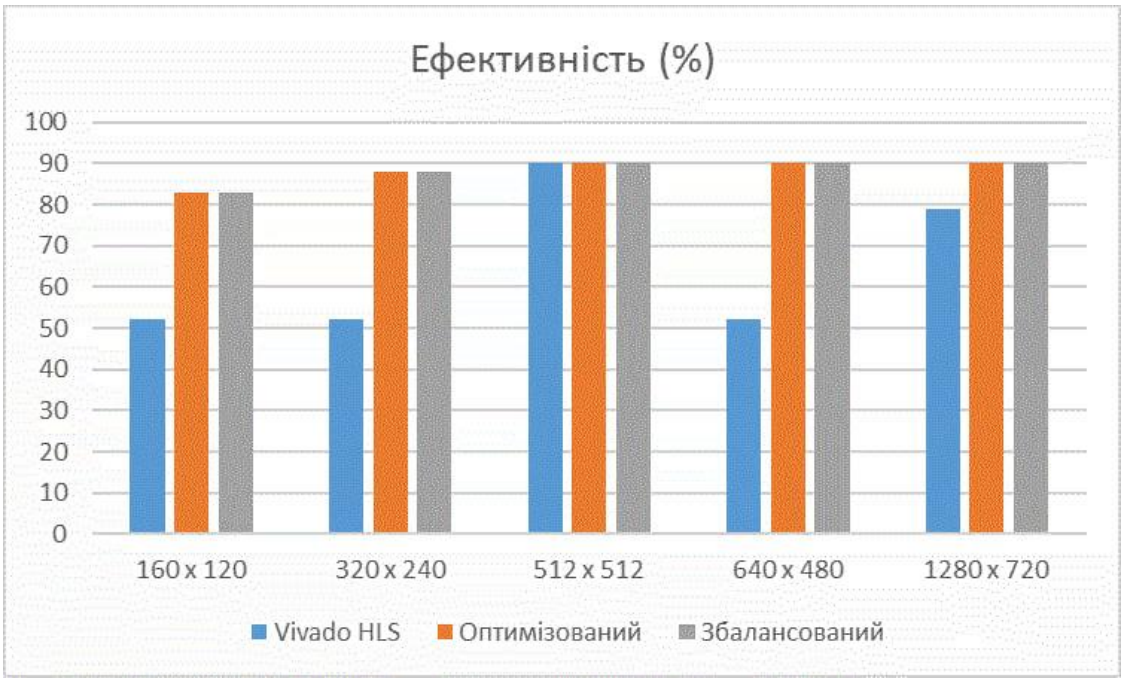


Рисунок 4.1 - Ефективність використання BRAM для кадрів RGB

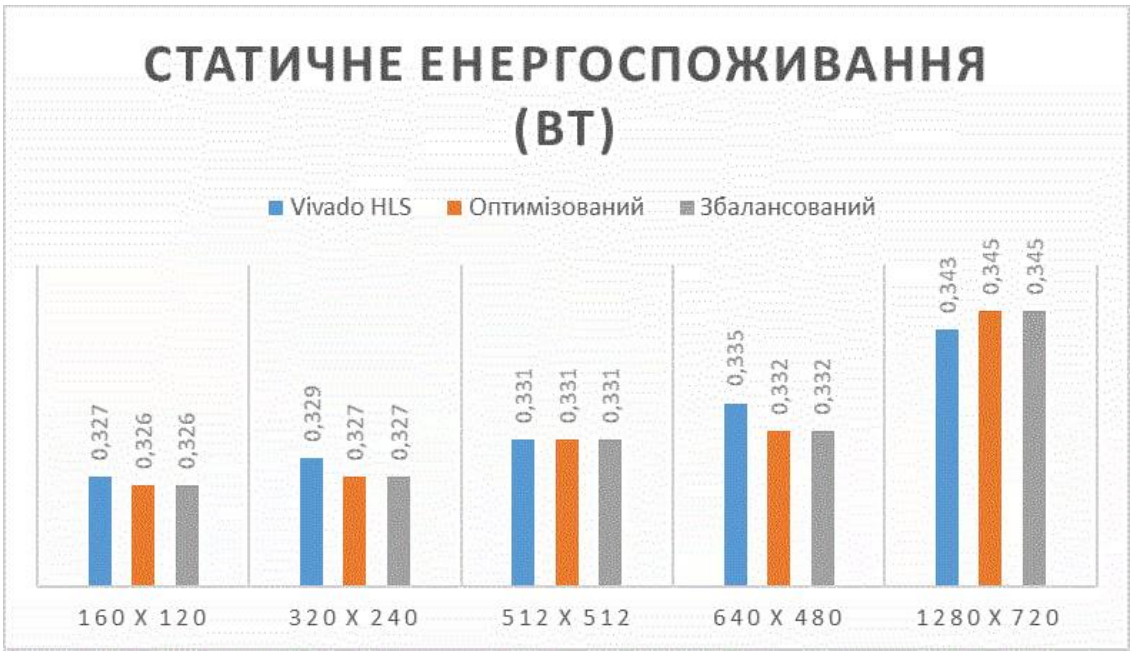


Рисунок 4.2 - Статичне енергоспоживання

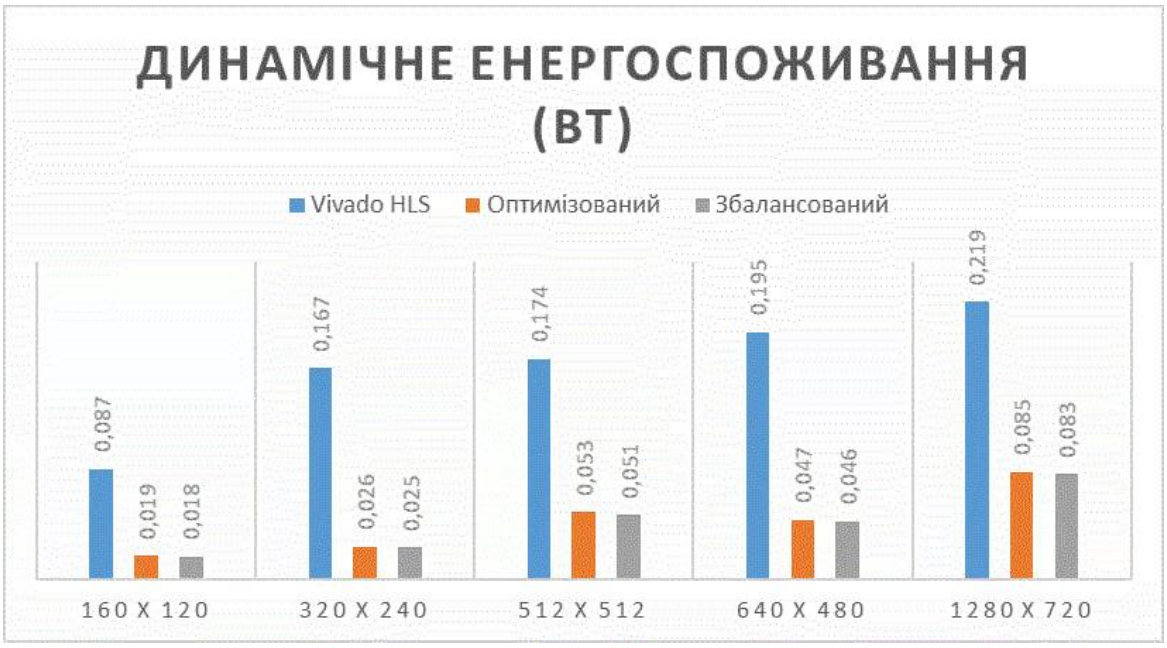


Рисунок 4.3 - Загальне динамічне енергоспоживання для послідовного читання/запису

Рисунок 4.4 - Загальне динамічне споживання електроенергії для ковзного вікна 3 × 3

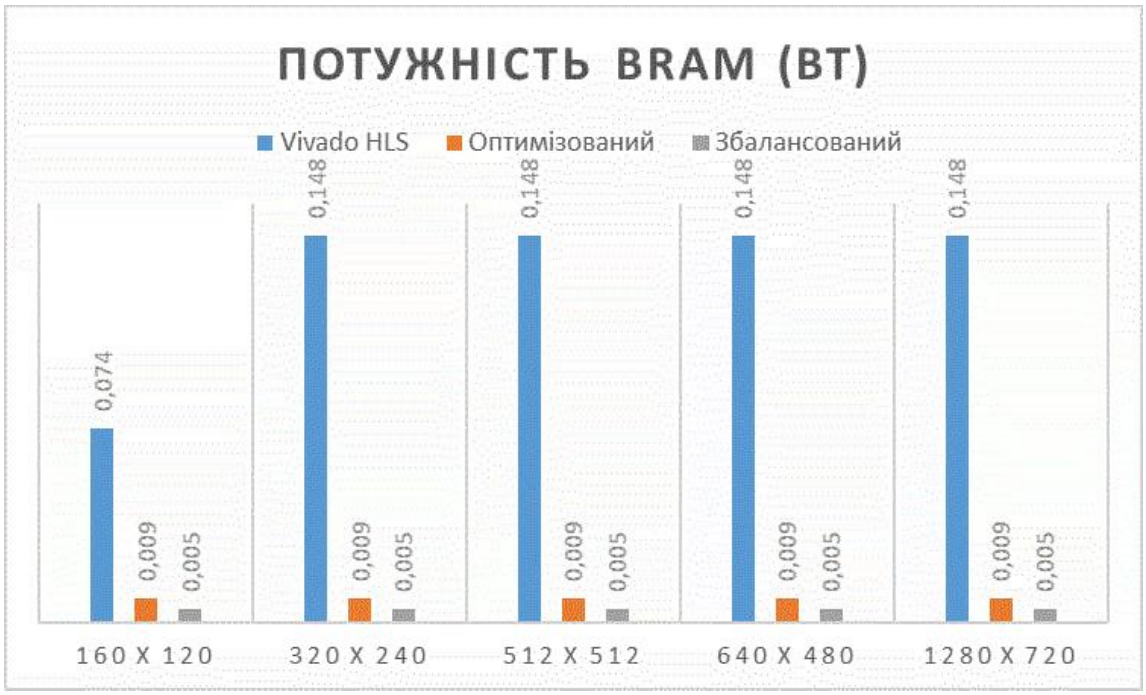


Рисунок 4.5 - Енергоспоживання BRAM для послідовного читання/запису

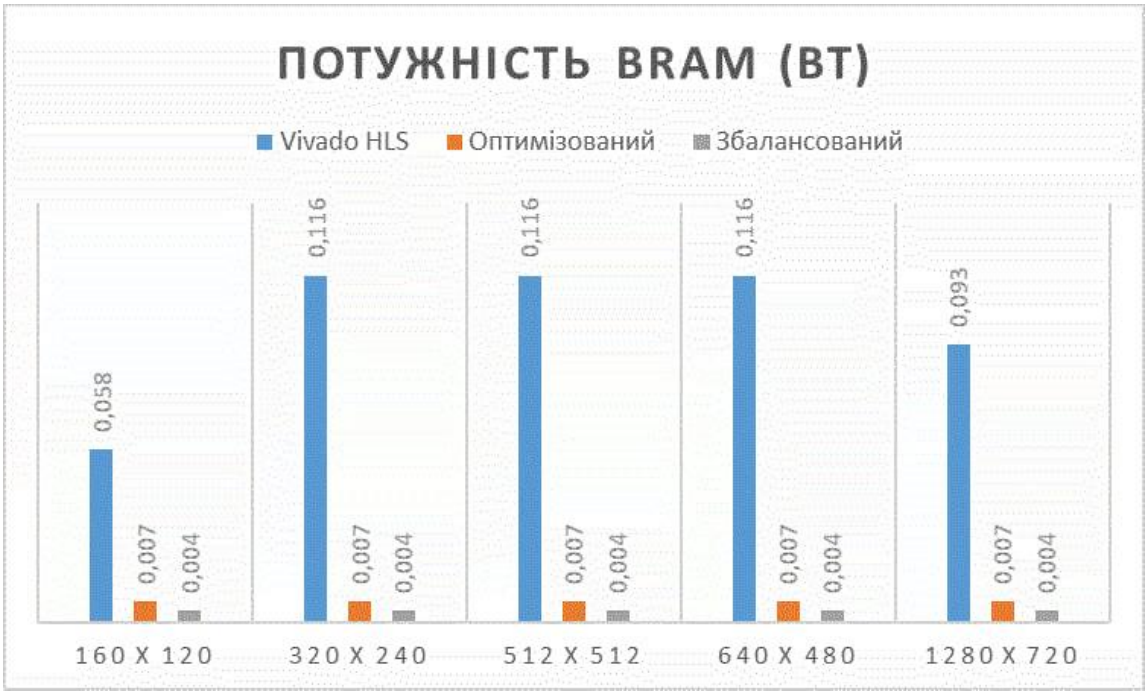


Рисунок 4.6 - Споживання енергії BRAM для зчитування ковзного вікна 3 × 3

4.3 Високорівнева обробка зображень

Третій набір дослідів контекстуалізує вплив розподілу пам’яті на системи обробки зображень високого рівня. Мета цього набору дослідів полягала в тому, щоб кількісно визначити, наскільки кадрові буфери впливають на використання ресурсів і енергоспоживання в повних системах обробки зображень на основі типових і запропонованих стратегій розділення.

Запропонований метод створювався на прикладах алгоритмів Optical Flow і MeanShift. Optical Flow оцінює видимий рух об’єктів, викликаний відносним рухом спостерігача; тобто для двох послідовних кадрів Optical Flow оцінює переміщення кожного пікселя (або більших областей) від одного кадру до іншого. Він належить до тимчасового класу алгоритмів обробки зображень, тобто виконує обчислення в часі (різні кадри). Реалізація Optical Flow базується на коді, доступному з [51], з використанням методу TV-L1, реконструйованого таким чином, щоб він відповідав вимогам синтезу Vivado HLS C (наприклад, динамічний розподіл пам’яті було замінено розподілом статичної пам’яті); інших оптимізацій не проводилось. Обчислюється один масштаб, а не кілька масштабів, для зображень розміром 160 × 120: приклад зображено на рисунку 4.7. Використовувався загальнодоступний набір даних із [44]. Було розроблено три версії: із розподілом пам’яті за замовчуванням і з дотриманням оптимізованого використання та збалансованих алгоритмів. Результати використання FPGA для Altera Cyclone V наведено в таблиці 4.3 (оптимізовані та збалансовані стратегії дають однакове використання BRAM, хоча різні конфігурації, для нашої реалізації). Для стратегії за замовчуванням блоків BRAM було недостатньо, щоб задовольнити всі вимоги до пам’яті, через що інструмент синтезу виводив LUT пам’яті для частин конструкції. Використовуючи даний метод, BRAM достатньо для реалізації повної системи. Споживана потужність для кожної версії зображена на рисунку 4.8.

(a) Перший кадр

(b) Другий кадр

(c) 1 шкала оптичного потоку

(d) 5 шкал оптичного потоку

Рисунок 4.7 - Результати Optical Flow. (a,b): вихідні кадри. (c): вихід з оптичного потоку 1 шкали (використовується в нашій реалізації FPGA). (d): вихід з 5 шкал оптичного потоку

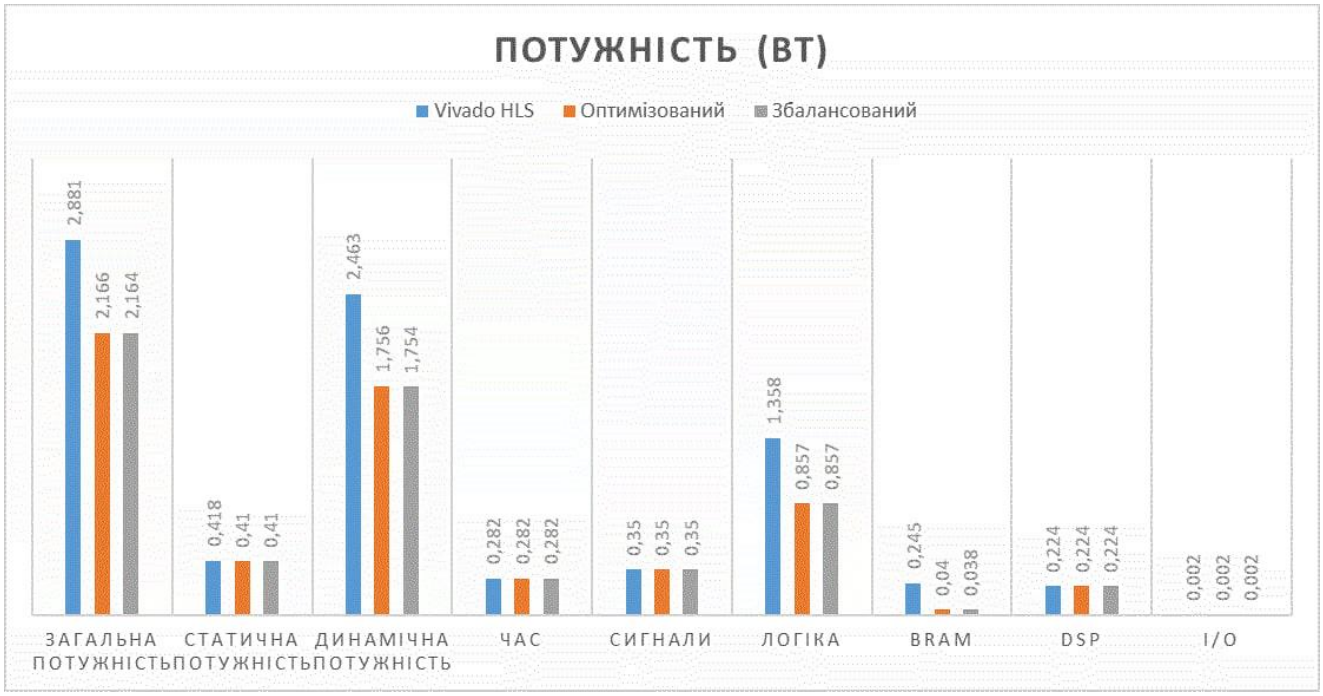


Рисунок 4.8 - Енергоспоживання TV-L1 Optical Flow на Cyclone V.

Алгоритм MeanShift [56] розраховує карту достовірності для положення об’єкта на зображенні на основі колірної гістограми такого об’єкта на попередньому зображенні: тобто для об’єкта, позиція якого відома, а кольорова гістограма обчислюється в кадрі k, алгоритм MeanShift визначає найвірогідніше положення об’єкта в кадрі k+1 на основі порівняння кольорової гістограми. Це часовий і динамічний алгоритм: він виконує обчислення в більш ніж одному кадрі, вимагаючи непередбачуваної кількості ітерацій (до попередньо визначеного максимуму) на непередбачуваних позиціях кадру (залежно від положення об’єкта під час виконання). Він був описаний на C і реалізований через Vivado HLS; дана реалізація була дуже оптимізована для апаратної реалізації. Відстеження MeanShift зберігає перший вхідний кадр (запис повного кадру в пам’ять у послідовному порядку старших рядків) і обчислює колірну гістограму області шириною M і висотою N, зосередженою на початковій позиції об’єкта (зчитування M × N пікселів) . Кожен наступний кадр зберігається, а кольорові гістограми для можливих нових позицій обчислюються в області навколо попередньої відомої позиції. Нова позиція визначається, коли різниця між попередньою та поточною позиціями є нижчою за заздалегідь визначену межу помилки або досягнуто максимальної кількості ітерацій. Шаблони доступу до відстеження MeanShift не є регулярними або передбачуваними, оскільки вони залежать від вхідних зображень; він є репрезентативним для алгоритмів обробки зображень із інтенсивним використанням пам’яті, оскільки результат залежить від повних (або непередбачуваних підмножин) сцен, а не від чітко визначених пікселів чи регіонів.

Система відстеження була реалізована на чіпові Zynq 7020 на Zedboard, підключеному до зовнішньої камери OV7670, див рисунок 4.9. Оброблені дані (зображення плюс положення відстежуваного об’єкта) надсилаються на бортовий процесор ARM, який повторно передає на віддалений робочий стіл через Ethernet. Однак важливо підкреслити, що для зв’язку та відображення повний алгоритм реалізовано на FPGA. На рисунку 4.10 показано сам ZedBoard.

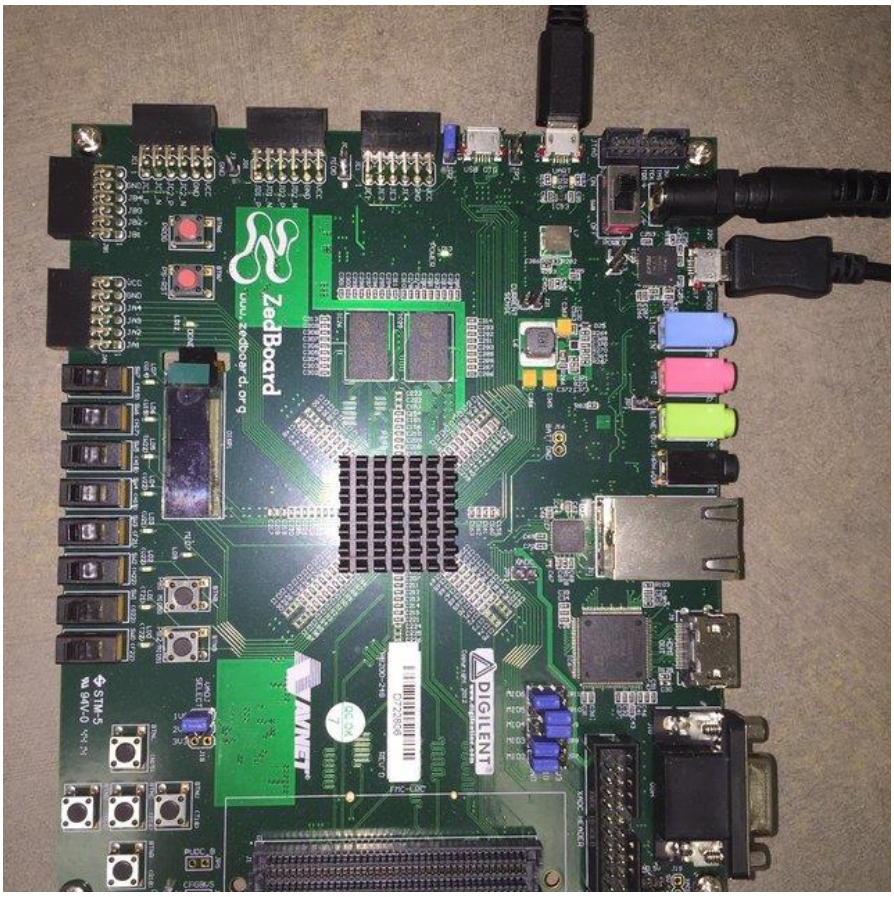


Рисунок 4.9 - Zedboard

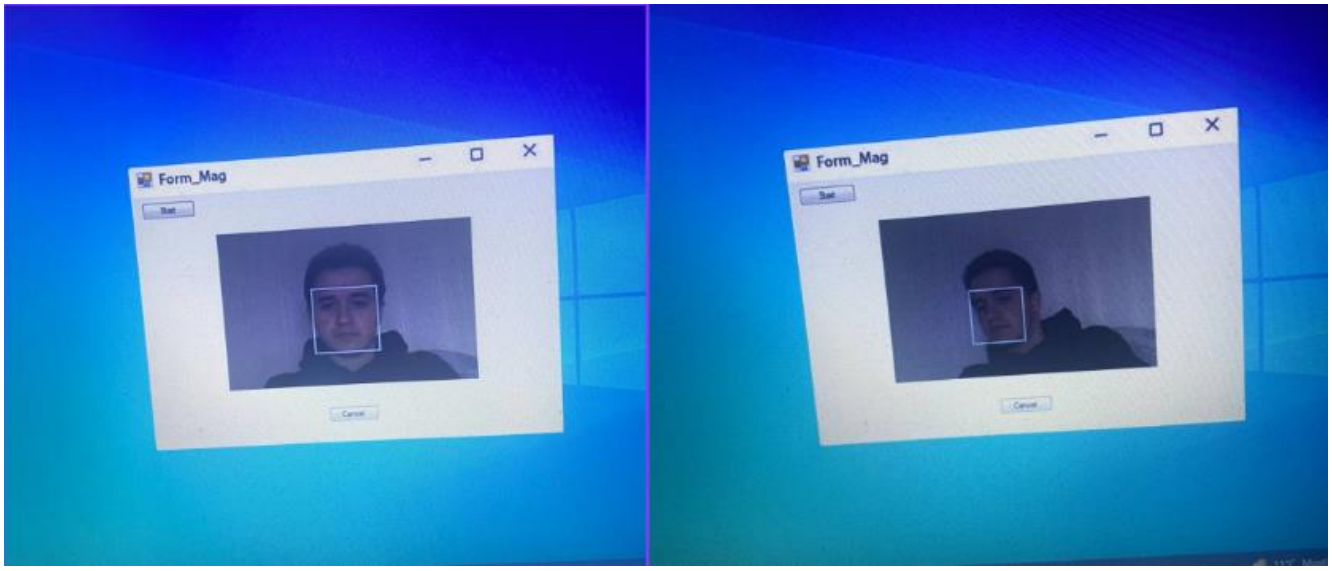


Рисунок 4.10 - Процес відстеження обличчя в реальному часі, що відображається на ПК. Зображення, надіслане з Zedboard через з’єднання Ethernet.

Було розроблено три версії системи: із розподілом пам’яті за замовчуванням, розподілом пам’яті з оптимізованим використанням і збалансованим розподілом для розмірів зображення 320 × 240, де кожен піксель має 24 біти (RGB), з областю інтересу розміром M = 16 і N = 21. Як і в попередньому експерименті, базовою лінією є реалізація відстеження MeanShift, згенерована Vivado HLS. Версії, які використовуються для порівняння, замінюють буфер кадрів HLS реалізаціями, закодованими вручну: усі інші модулі відстеження MeanShift не змінені (генеруються від C до Vivado HLS). Використання ресурсів для кожної версії зображено в таблиці 4.3. Споживання електроенергії для кожної версії зображено на рисунку 4.11.

Таблиця 4.4 – MeanShift – використання ресурсів і продуктивність FPGA: створено з Vivado HLS проти модифікацій, закодованих вручну, відповідно до запропонованих алгоритмів.

	Розподіл пам’яті за замовчуванням	Розподіл пам’яті з опимізованим використанням	Збалансований розподіл пам’яті
FF	6264 (5%)	6264 (5%)	6264 (5%)
LUTs	9197 (17%)	9310 (17.5%)	9475 (17.8%)
IOs	62 (32%)	62 (32%)	62 (32%)
BRAM	228 (81%)	150 (54%)	150 (54%)
DSPs	8 (3%)	8 (3%)	8 (3%)
fps	134	134	134

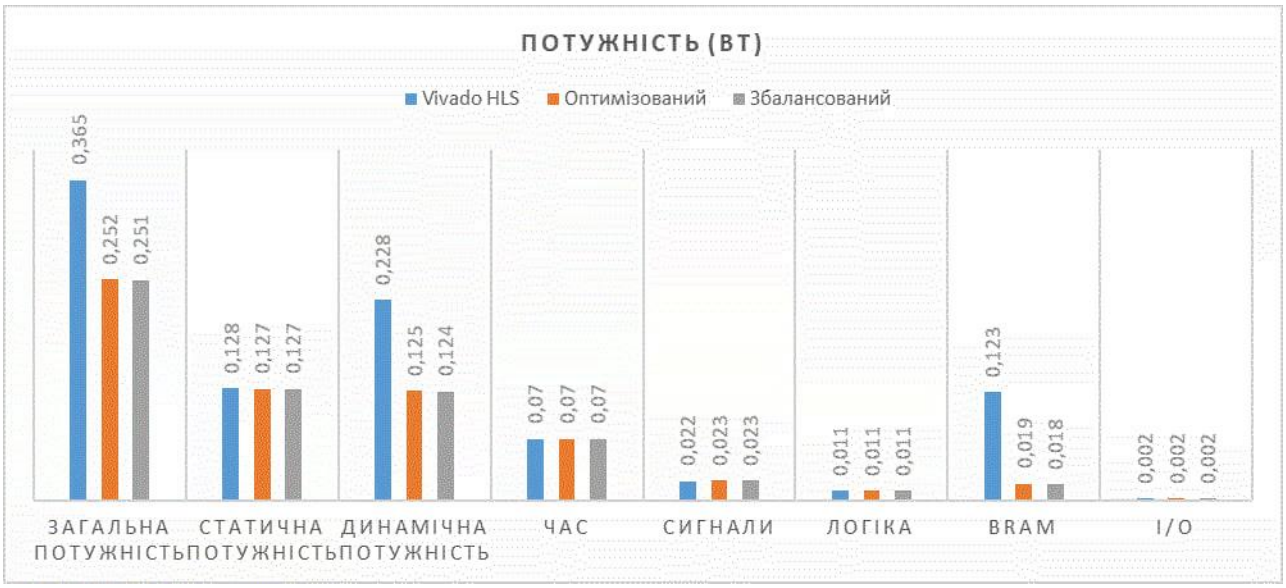


Рисунок 4.11 - MeanShift Відстеження енергоспоживання на Zedboard

4.4 Огляд результатів роботи

Вибір оптимальних конфігурацій для задачі на обробку зображень залежить від розміру зображення та бажаної ефективності. Наприклад, якщо розмір зображення 320 × 240, різні конфігурації можуть мати різний рівень ефективності та різне споживання енергії. У такому випадку, необхідно знайти оптимальну конфігурацію, яка забезпечує баланс між ефективністю та споживанням енергії.

Однак, якщо розмір зображення збільшується до 512 × 512, ефективність однакова для всіх конфігурацій, але споживання енергії все ще може відрізнятися. Це пов'язано з тим, що при збільшенні розміру зображення збільшується кількість операцій, які необхідно виконати, тому споживання енергії може збільшитися.

Також, становище конфігурацій BRAM 9 × 2048 та 4 × 4096 демонструє, що іноді менш енергоефективні конфігурації можуть досягати однакової ефективності порівняно з більш енергоефективними конфігураціями. Це пов'язано зі складністю логіки, яка залежить від кількості та розміру модулів використаних у конфігурації. В такому випадку необхідно зважати на рівень ефективності та споживання енергії, щоб знайти оптимальну конфігурацію для конкретної задачі.

Крім того, важливо зазначити, що при виборі конфігурацій для певної задачі

потрібно враховувати не тільки потужність та ефективність, а й інші параметри, такі як об’єм пам’яті, швидкість обчислень, масштабованість та інші. Наприклад, якщо задача вимагає більшої кількості пам’яті, то можна вибрати конфігурацію з більшим обсягом пам’яті, навіть якщо це приведе до збільшення споживання енергії. З іншого боку, якщо задача потребує високої швидкості обчислень, то можна вибрати конфігурацію з більш потужним процесором, навіть якщо це призведе до збільшення споживання енергії.

Також, варто враховувати, що зазвичай найбільш енергоефективними є конфігурації з використанням спеціалізованих пристроїв, таких як графічні процесори (GPU) та тензорні процесори (TPU), які спеціально розроблені для роботи з великими обсягами даних та інтенсивними обчисленнями. Ці пристрої мають вищу потужність та ефективність, ніж загальнопризначені процесори, тому можуть бути кращим вибором для деяких задач.

Отже, вибір конфігурацій для вирішення певної задачі потребує компромісу між різними параметрами, такими як потужність, ефективність, об’єм пам’яті та швидкість обчислень. Врахування цих параметрів може допомогти забезпечити максимальну ефективність та економію енергії при вирішенні задачі.

Досліди показують, що використані для вирішення задачі алгоритми розподілу досягають вищої ефективності, ніж стандартні стратегії синтезу, за винятком кадрів розміром 512 × 512, де ефективність не змінюється. Це той випадок, коли стратегії за замовчуванням працюють однаково добре з точки зору використання, оскільки висота та ширина зображення є степенями 2. Це підтверджує, що для покращення використання пам’яті потрібні змінені стратегії розділення відповідно до вимог.

Статичне енергоспоживання, зменшується залежно від розміру кадру, за винятком кадрів розміром 512 × 512 і 1280 × 720, де різниця в ефективності використання між стратегіями за замовчуванням і запропонованими є найменшими, а додаткова логіка адресації стає занадто (статичною). Це підтверджує, що проблеми використання та живлення неподільні, і їх потрібно розглядати спільно.

Загальна динамічна потужність в експериментах, проведених на буферах кадрів, зменшується в середньому на 74,708% (σ = 7,819%) для читання/запису і в середньому на 72,206% (σ = 12,546%) для лише читання. Це підтверджує гіпотезу про те, що розділення пам’яті надає можливості для зменшення енергоспоживання, незважаючи на потребу в логічних витратах. Враховуючи лише динамічну потужність BRAM, використані для вирішення задачі методи розподілу призводять до зниження середньої потужності на 95,945% (σ = 1,351%) для дослідів із читанням/записом і на 95,691% зниження середньої потужності (σ = 1,331%) для дослідів лише з читанням.

У дослідах із використанням Optical Flow, де потужність BRAM і LUT пам’яті становить 24,3% загального споживання енергії та 32% динамічної потужності, можна побачити, що запропоновані алгоритми розподілу можуть зменшити загальну потужність приблизно на 25%. Для відстеження MeanShift, де на потужність BRAM припадає 34,55% загального енергоспоживання та 55,91% динамічної потужності, можна побачити, що запропоновані алгоритми розподілу можуть зменшити загальну потужність приблизно на 30%. Продуктивність алгоритму (тобто кількість кадрів за секунду) не вплинула на методології розподілу як в Optical Flow, так і в MeanShift, оскільки стратегії використані для вирішення задачі - не впливають на затримки доступу до пам’яті, а максимальні тактові частоти залишаються незмінними (буфери кадрів не відповідали за критичний шлях синхронізації). Результати роботи вигідно відрізняються від результатів, наведених у [28], де досягнуто зниження потужності BRAM до 26% за рахунок зниження тактової частоти на 1,6%; методологія, яка використовувалась для вирішення задачі забезпечує зниження потужності BRAM до 74% без шкоди для тактової частоти. Це пов’язано з тим, що їхній підхід не враховує відмінності в енергоспоживанні, спричинені різними конфігураціями BRAM, що є ключовим аспектом даної методології.

4.5 Споживання енергії

Також було проілюстровано, як різні конфігурації BRAM впливають на енергоспоживання: залежно від того, скільки BRAM має бути стробовано, щоб отримати доступ до пікселя (іншими словами, залежно від того, яка конфігурація використовується для розподілу пам’яті), витрачаються різні кванти енергоспоживання. (припускаючи, що решта закриті, згідно з методологією застосованою для вирішення задачі). В даній роботі було проілюстровано, як за однакового розміру RAM різні конфігурації можуть зменшити енергоспоживання, що витрачається на BRAM, до 82%; це відповідало загальному динамічному зниженню потужності до 50%. Ці результати показали, наскільки сильно конфігурація BRAM впливає на енергоспоживання. Варто зауважити, що дуже невелике зменшення використання BRAM (тобто кількості BRAM, необхідних для реалізації кадрового зберігання) може призвести до значного зниження потужності.

У проведених в цій роботі дослідах із використанням складних високорівневих алгоритмів було показано, що потужність BRAM становить значну частину загального споживання енергії: а саме, використовуючи стратегію Vivado HLS за замовчуванням, на BRAM припадає 8% споживання енергії Optical Flow і 34% споживання енергії Meanshift. Крім того, значна потужність витрачається на логіку через зміну виходу BRAM (запобігання цього в підході для вирішення задачі - відбувається через стратегії синхронізації).

4.6 Висновки

Описаний в роботі підхід до виділення пам'яті, який був описаний в попередніх розділах, є більш ефективним з точки зору використання ресурсів FPGA. Оскільки завдяки мінімізації використання пам'яті, збільшується кількість логіки, що може бути реалізована на даному FPGA, що забезпечує більшу продуктивність і швидкість обробки даних.

В цьому методі використовується більш складна адресацію, що дозволяє

використовувати різні конфігурації BRAM для керування пам'яттю, такі як сигнали дозволу запису та декодування адреси. Це збільшує складність логіки керування пам'яттю і може призвести до більшого використання LUT. Однак, дослідження показали, що ці накладні витрати LUT є досить незначними, якщо порівнювати з позитивним ефектом, що має наш підхід на продуктивність та швидкість обробки даних.

Більш того, цей метод дає можливість змінювати стратегії розділення відповідно до вимог і налаштувань, що робить його гнучкішим та більш універсальним для використання в різних застосуваннях та сценаріях. Вважається, що цей метод до виділення пам'яті може бути корисним для багатьох інженерних проектів, де ефективність ресурсів FPGA є критично важливою. У дослідах, проведених в даній роботі із використанням високорівневих алгоритмів (відстеження середнього зсуву та оптичного потоку) ці накладні витрати LUT становили 0,8 і 2,0 відсоткових пункти відповідно.

В даному розділі було реалізовано метод побудови архітектури вбудованих систем обробки зображень на основі FPGA.

ВИСНОВКИ

Ефективне зіставлення високорівневих описів кадрів зображень із низькорівневими системами пам’яті є важливим фактором для широкого впровадження FPGA як платформ для розгортання програм обробки зображень високого рівня. Алгоритми розділення є одним із методів проектування, який забезпечує шляхи до енергоефективних проектів, які можуть відповідати вимогам сучасних додатків.

На основі формалізації параметрів конфігурації BRAM і моделі потужності пам’яті було продемонстровано, як алгоритми розділення можуть перевершити традиційні стратегії в контексті синтезу високого рівня. Дії проведені для вирішення задачі показують, що запропоновані алгоритми можуть призвести до підвищення ефективності використання на 60%, збільшення розмірів і/або кількості кадрів, які можна розмістити на чіпі, і зменшити динамічне споживання енергії буферами кадрів приблизно до 70%. У проведених дослідах із використанням Optical Flow і MeanShift, репрезентативних високорівневих алгоритмів обробки зображень, дані показують, що алгоритми розподілу можуть зменшити загальну потужність до 25% і 30% відповідно без будь-якого зниження продуктивності. Наші стратегії можна застосувати до будь-якої сімейства FPGA і легко масштабувати за потреби для майбутніх платформ FPGA з новими можливостями та конфігураціями пам’яті чіпа.

Більшість методів проектування HLS зосереджено на програмованості та продуктивності.

В роботі були розглянуті питання, пов'язані з підходами до проектування апаратного забезпечення на основі FPGA, моделями потоків даних і паралельними обчисленнями, а також дається огляд їх передумов і пов'язаних з ними робіт.

Для реалізації підвищення швидкодії систем обробки зображень на основі FPGA було проведено:

1. дослідження методів побудови архітектури вбудованих систем обробки зображень на основі FPGA;

2. аналіз сучасних програмно-технічних засобів для обробки зображень на основі FPGA

3. розробка моделі функціонування програмно-технічних засобів обробки зображень на основі FPGA;

4. розробка методу побудови архітектури вбудованих систем обробки зображень на основі FPGA;

5. реалізація методу побудови архітектури вбудованих систем обробки зображень на основі FPGA.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. 1. Лисенко С.М., Ванярха О.С., Бондарук О.В. Метод побудови архітектури вбудованих систем обробки зображень на основі FPGA. *Вісник Хмельницького національного університету*. 2023.№3.

2. Satyanarayanan M., Simoens P., Xiao Y., Pillai P., Chen Z., Ha K., Hu W., Amos B. Edge Analytics in the Internet of Things. *IEEE Pervasive Computing*. 2015. vol. 14, no. 2, pp. 24 – 31.

3. Gandomi A. and Haider M. Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management (IJIM).* 2015. vol. 35, no. 2, pp. 137-144.

4. Video Analytics Hardware, Software, and Services Revenue to Reach $3 Billion by 2022. URL: https://www.embedded-vision.com/industry-analysis/market-analysis/ video-analytics-hardware-software-and-services-revenue-reach-3-bil.

5. I. L. Markov, “Limits on Fundamental Limits to Computation,” Coputing Research Laboratory. URL: http://arxiv.org/abs/1408.3821

6. Programmable embedded platforms for remote and compute intensive image processing applications. URL: http://gow.epsrc.ac.uk/NGBOViewGrant.aspx?GrantRef=EP/K009583/1

7. Stewart R., Duncan K., Michaelson G., Garcia P., Bhowmik D., Wallace A. RIPL: A Parallel Image Processing Language for FPGAs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*. 2018. vol. 11, no. 1, Mar., pp. 1-28.

8. High-level synthesis of dataflow programs for heterogeneous platforms. URL: https://infoscience.epfl.ch/record/207992/files/EPFLTH6653.pdf

9. Bacon D.F., Rabbah R., Shukla S. FPGA Programming for the Masses. *ACM Queue Magazine*. 2013. vol. 11, no. 2, pp. 40-52.

10. Gort M., Anderson J. Design Re-use for Compile Time Reduction in FPGA High-level Synthesis Flows. *Proceedings of IEEE International Conference on Field-Programmable Technology (FPT)*. 2014. pp. 4-11.

11. Stitt G., Coole J. Intermediate Fabrics: Virtual Architectures for Near-Instant FPGA Compilation. *IEEE Embedded Systems Letters*. 2011. vol. 3, no. 3, pp. 81-84.

12. Baxter R., Booth S., Bull M., Cawood G., D'Mellow K., Guo X., Parsons M., Perry J., Simpson A., Trew A. High-Performance Reconfigurable Computing - the View from Edinburgh. *Proceedings of the Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. 2007. pp. 373-279.

13. Chenini H., D´erutin J.P., Aufrère R., Chapuis R. Parallel Embedded Processor Architecture for FPGA-based Image Processing using Parallel Software Skeletons. *EURASIP Journal on Advances in Signal Processing*. 2013. vol. 2013, no. 1, p. 153.

14. Natvig L., Iordan A., Eleyat M., Jahre M., Amundsen J. Multi- and Many-Cores, Architectural Overview for Programmers. *IEEE Embedded Systems Letters.* 2017. ch. 1, pp. 1-27.

15. Vajda A. Multi-core and Many-core Processor Architectures. *IEEE Embedded Systems Letters.* 2011. ch. 2, pp. 9-43.

16. Soft GPGPUs for Embedded FPGAs: An Architectural Evaluation. URL: http://arxiv.org/abs/1606.06454

17. Kapre N. Custom FPGA-based Soft-processors for Sparse Graph Acceleration. *Proceedings of IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. 2015. pp. 9-16.

18. Lieske T., Reichenbach M., Ringlein B., Fey D. Dataflow Optimization for Programmable Embedded Image Preprocessing *Accelerators. Proceedings of IEEE International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. 2016. pp. 1-8.

19. Wang H., Li Y., Qian J. Design and Implementation of an FPGA-Based Real-Time Video Surveillance System. *Journal of Real-Time Image Processing*. 2019. vol. 16, no. 3, pp. 723-734.

20. Puschner P., Rocha A., Silva C., Spars J., Tocchi A. T-CREST: Time-

predictable Multi-core Architecture for Embedded Systems. *Journal of Systems Architecture (JSA)*. 2015. vol. 61, no. 9, pp. 449-471.

21. Heisswolf J., Zaib A., Weichslgartner A., Karle M., Singh M., Wild T., Teich J., Herkersdorf A., Becker J. The invasive network on chip - a multi-objective many-core communication infrastructure. In: Proc. *IEEE International Workshop on Architecture of Computing Systems (ARCS)*. 2014, pp. 1-8.

22. She D., He Y., Waeijen L., Corporaal H. A Co-Design Framework with OpenCL Support for Low-Energy Wide SIMD Processor. *Journal of Signal Processing Systems (JSPS)*. 2015. vol. 80, no. 1, pp. 87-101.

23. Fernando S., Siyoum F., He Y., Kumar A., Corporaal H. MAMPSx: A design framework for rapid synthesis of predictable heterogeneous MPSoCs. In: Proc. *IEEE International Symposium on Rapid System Prototyping (RSP)*. 2013, pp. 136-142.

24. S´erot J., Berry F., Bourrasset C. High-level dataflow programming for real-time image processing on smart cameras. *Journal of Real-Time Image Processing (JRTIP)*. 2016. vol. 12, no. 4, pp. 635-647.

25. Liu C. A rapid FPGA loop accelerator design framework using soft CGRA overlay. *IEEE International Conference on Field Programmable Technology (FPT)*. 2015. Pp. 56-63.

26. Satyanarayanan M., Simoens P., Xiao Y., Pillai P., Chen Z., Ha K., Hu W., Amos B. Edge Analytics in the Internet of Things. *IEEE Pervasive Computing*. 2015. vol. 14, no. 2, pp. 24-31.

27. Chen D., Chen H., Keckler S.W. Designing FPGA Accelerators for Machine Learning Applications. *IEEE Micro*. 2018. vol. 38, no. 2, pp. 32-40.

28. Cong J., Liu B., Neuendorffer S. High-Level Synthesis for FPGAs: From Prototyping to Deployment. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2011. vol. 30, no. 4, pp. 473-491.

29. Farhan M., Sidek O.M. A Review of Embedded System Designs Using Field Programmable Gate Arrays. *International Journal of Engineering and Technology (IJET)*. 2015. vol. 7, no. 3, pp. 235-243.

30. Mishra P., Rao K.R.K., Shastry G.V.R. High Speed Implementation of

Image Processing Algorithms on FPGA. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE)*. 2013. vol. 2, no. 11, pp. 575-576.

31. Joshi M.V., Rathore S.S. Implementation of Image Processing Algorithms on FPGA: A Review. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE)*. 2013. vol. 2, no. 11, pp. 577-577.

32. Ali A.N.N., Selvarani R. Hardware Implementation of Image Processing Algorithms on FPGA: A Review. *International Journal of Computer Science and Mobile Computing (IJCSMC)*. 2013. vol. 2, no. 5, pp. 56-64.

33. Varghese J.N., Varghese R.J. A Survey of FPGA Based Image Processing. *International Journal of Emerging Technology and Advanced Engineering (IJETAE)*. 2012. vol. 2, no. 6, pp. 329-333.

34. Das R.S., Sahu S. Image Processing on FPGA: A Review. International *Journal of Advanced Research in Computer Engineering and Technology (IJARCET)*. 2013. vol. 2, no. 10, pp. 346-346.

35. Zhang X., Lin X., Albonesi D.H. FPGA Acceleration of Convolutional Neural Networks for Wireless Visual Sensor Networks. *IEEE Transactions on Computers*. 2018. vol. 67, no. 5, pp. 722-735.

36. Bu F., Cong J., Liu S., Wen W. Efficient FPGA Implementation of CNNs with Variable Filter Sizes. *ACM Transactions on Embedded Computing Systems (TECS)*. 2017. vol. 16, no. 3, pp. 1-22.

37. Yalcin S.E., Yazici B., Cicekoglu O. A Real-Time Image Processing System on FPGA. *IEEE Transactions on Consumer Electronics. 2015. vol*. 61, no. 2, pp. 184-191.

38. Zhang C., Li P., Sun G., Guan Y., Xiao B., Cong J. Optimizing FPGA-Based Accelerator Design for Deep Convolutional Neural Networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2017. vol. 36, no. 7, pp. 114-115.

39. Anderson J.A., Willsky A.S. Digital Image Processing Using Field-

Programmable Gate Arrays. *IEEE Transactions on Image Processing*. 1997. vol. 6, no. 10, pp. 135-163.

40. Chen Y., Luo T., Liu S., Zhang S., He L., Wang J., Li L., Chen T. DNNBuilder: An Automated Tool for Building High-Performance DNN Hardware Accelerators for FPGAs. *ACM Transactions on Embedded Computing Systems (TECS)*. 2018. vol. 17, no. 2, pp. 1-24.

41. Wang X., Tang H., Chen M., Wei W., Zhang C. FPGA-Based Edge Detection for Real-Time Vision Systems. *IEEE Transactions on Circuits and Systems for Video Technology*. 2018. vol. 28, no. 11, pp. 209-221.

42. Singh S., Tripathi S., Gupta S.C., Raj B. Efficient Implementation of Image Processing Algorithm on FPGA for Traffic Sign Recognition. *Journal of King Saud University - Computer and Information Sciences*. 2017. vol. 29, no. 4, pp. 490-496.

43. Bithe D.D., Mali M.B., Thombare V.D. FPGA-Based Image Processing: A Review. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT)*. 2018. vol. 3, no. 1, pp. 507-514.

44. Timmapur S.R., Rao S.S., Rao N.N. FPGA Based Real Time Edge Detection Techniques for Image Processing Applications. *Journal of VLSI Design Tools and Technology (JoVDTT)*. 2016. vol. 6, no. 1, pp. 1-7.

45. Li X., Wang Y., Chen L., Li H., Zhang S. Implementation of an Improved Median Filter Algorithm Based on FPGA. *Journal of Physics: Conference Series*. 2018. vol. 1000, no. 1, pp. 1-7.

46. Niu K., Li J., Zhao J. Design of FPGA-Based Edge Detection System for Real-Time Image Processing. *Journal of Signal Processing Systems*. 2018. vol. 92, no. 6, pp. 883-892.

47. Hong S., Jung S., Kim S., Park C. FPGA-Based Acceleration of Convolutional Neural Networks Using Winograd Algorithm. *IEEE Transactions on Circuits and Systems for Video Technology*. 2020. vol. 30, no. 5, pp. 35-40.

48. Gao H., Wu X., Zhao Q., Jin X., Zhang H. An FPGA-Based Real-Time Image Processing System for Infrared Small Target Detection. *IEEE Access*. 2019. vol.

7, pp. 42-53.

49. Chang L., Zhang Y., Cheng Y., Chen J., Liu H. FPGA-Based Parallel Architecture for Efficient Image Processing. *IEEE Transactions on Circuits and Systems for Video Technology*. 2018. vol. 28, no. 6, pp. 42-53.

50. Li S., Li B., Li X. FPGA-Based Efficient Real-Time Stereo Matching for Large-Scale Images. *IEEE Transactions on Circuits and Systems for Video Technology*. 2020. vol. 30, no. 6, pp. 38-49.

51. Han J., Oh M. FPGA-Based Real-Time Image Processing System for Object Detection and Tracking. *IEEE Transactions on Consumer Electronics*. 2017. vol. 63, no. 2, pp. 111-117.

52. Akkaladevi S., Das D. FPGA-Based Real-Time Image Processing System for Surveillance Applications. *IEEE Transactions on Circuits and Systems for Video Technology*. 2017. vol. 27, no. 10, pp. 15-26.

53. Wang B., Yang X., Chen G. A Scalable FPGA-Based System for Real-Time Image Processing. *IEEE Transactions on Circuits and Systems for Video Technology*. 2017. vol. 27, no. 11, pp. 68-79.

54. Subramanian P., Sengupta S. FPGA-Based Real-Time Image Processing for Unmanned Aerial Vehicle Navigation. *IEEE Transactions on Aerospace and Electronic Systems*. 2016. vol. 52, no. 5, pp. 198-211.

55. Chen R., Zhang B., Zhou Z., Chen J., Yao Y. FPGA-Based High-Speed Image Processing System for Microscope Applications. *IEEE Transactions on Industrial Electronics*. 2018. vol. 65, no. 5, pp. 32-41.

56. Dong R., Xu J., Jin X., Liu W. A Novel FPGA-Based System for Real-Time Video Processing. *Journal of Signal Processing Systems*. 2017. vol. 88, no. 3, pp. 89-100.

57. Wang H., Li Y., Qian J. Design and Implementation of an FPGA-Based Real-Time Video Surveillance System. *Journal of Real-Time Image Processing*. 2019. vol. 16, no. 3, pp. 23-34.

58. Shi T., Xie Y., Wang Y., Xue H., Chen Y. An FPGA-Based Real-Time Image Processing System for Road Traffic Surveillance. *IEEE Access*. 2020. vol. 8, pp. 81-92.

59. Zhang W., Qian J., Yu J. FPGA-Based Real-Time Image Processing System for Road Marking Recognition. *Journal of Real-Time Image Processing*. 2018. vol. 14, no. 1, pp. 131-139.

60. Xu J., Liu Y., Jin X., Dong R. FPGA-Based Real-Time Image Processing System for Line Detection in Tunnel Inspection. *IEEE Transactions on Industrial Informatics*. 2018. vol. 14, no. 1, pp. 17-25.

61. Kim H., Kang H., Lee S. FPGA-Based Real-Time Image Processing for High-Resolution Mobile Display. *Journal of Display Technology*. 2017. vol. 13, no. 12, pp. 54-60.

62. Kim J., Kang H., Lee S. FPGA-Based Real-Time Image Processing for Object Detection in UAVs. *Journal of Intelligent and Robotic Systems*. 2018. vol. 92, no. 1, pp. 61-74.

63. Zhang Y., Xu X., Zhang Z. FPGA-Based Real-Time Image Processing System for Object Recognition. *Journal of Real-Time Image Processing*. 2016. vol. 12, no. 3, pp. 81-91.

64. Huang J., Chen Y., Kuo S. Design of an FPGA-Based Real-Time Image Processing System for Biomedical Applications. *Journal of Medical and Biological Engineering*. 2019. vol. 39, no. 5, pp. 67-70.

65. He X., Liu X., Chen J., Yu J., Xue Y. FPGA-Based Real-Time Image Processing System for Robot Vision. *Journal of Real-Time Image Processing*. 2018. vol. 13, no. 2, pp. 29-38.

66. Zhou C., Yang X., Zhang K. An FPGA-Based Real-Time Image Processing System for Vehicle License Plate Recognition. *Journal of Real-Time Image Processing*. 2019. vol. 16, no. 6, pp. 21-35.

67. Afzali-Kusha A., Ashourian M. An FPGA-Based Real-Time Image Processing System for 3D Object Recognition. *International Journal of Signal Processing, Image Processing and Pattern Recognition*. 2016. vol. 9, no. 5, pp. 53-62.

68. Zhang S., Tang J., Li J., Liu S. Design and Implementation of an FPGA-Based Real-Time Image Processing System for Moving Target Detection. *International*

Journal of Distributed Sensor Networks. 2019. vol. 15, no. 8, pp. 1-12.

69. Chen Y., Jiang J., Kuo S. An FPGA-Based Real-Time Image Processing System for Visual Inspection of Printed Circuit Boards. *Journal of Electronic Testing*. 2020. vol. 36, no. 5, pp. 21-32.

70. Li X., Yao C., Zhang X., Li D. An FPGA-Based Real-Time Image Processing System for Facial Expression Recognition. *Journal of Real-Time Image Processing*. 2018. vol. 13, no. 4, pp. 59-69.

71. Li S., Song Y., Yan B. A Low-Power FPGA-Based Real-Time Image Processing System for Object Tracking. *Journal of Signal Processing Systems*. 2017. vol. 87, no. 2, pp. 27-37.

72. Wang Y., Huang H., Cheng Y., Lin Y. An FPGA-Based Real-Time Image Processing System for Traffic Sign Detection and Recognition. *Journal of Real-Time Image Processing*. 2018. vol. 14, no. 3, pp. 39-49.

73. Cao C., Guo Q., Du Y. Design and Implementation of an FPGA-Based Real-Time Image Processing System for Fruit Classification. *Journal of Real-Time Image Processing*. 2019. vol. 15, no. 2, pp. 41-45.

74. Li T., Huang Z., Xue L. An FPGA-Based Real-Time Image Processing System for Autonomous Navigation of Micro Aerial Vehicles. *Journal of Intelligent and Robotic Systems*. 2018. vol. 91, no. 1, pp. 21-35.

75. Fan S., Liu S., Zhang C. An FPGA-Based Real-Time Image Processing System for Face Recognition in Access Control. *Journal of Real-Time Image Processing*. 2016. vol. 12, no. 4, pp. 37-48.

76. Huang W., Zhou Y., Yang Y. An FPGA-Based Real-Time Image Processing System for Gesture Recognition. *Journal of Real-Time Image Processing*. 2021. vol. 18, no. 2, pp. 61-72.

77. Liu Y., Liu Q., Liu J. An FPGA-Based Real-Time Image Processing System for Human Action Recognition. *Journal of Real-Time Image Processing*. 2020. vol. 17, no. 4, pp. 67-79.

78. Chen S., Luo W., Deng Y. An FPGA-Based Real-Time Image Processing System for Stereo Vision. *International Journal of Distributed Sensor Networks*. 2018.

vol. 14, no. 2, pp. 1-13.

79. Wu Y., Wang X., Liu Y. An FPGA-Based Real-Time Image Processing System for Color Detection. *Journal of Real-Time Image Processing*. 2019. vol. 16, no. 6, pp. 67-78.

80. Zhang C., Chen Y., Shi Y. An FPGA-Based Real-Time Image Processing System for Target Tracking. *Journal of Real-Time Image Processing*. 2019. vol. 15, no. 1, pp. 67-80.

81. Wu J., Wu X., Chen S. An FPGA-Based Real-Time Image Processing System for Medical Image Segmentation. *Journal of Real-Time Image Processing*. 2019. vol. 15, no. 3, pp. 59-71.

ДОДАТОК А
(обов'язковий)

КОПІЯ ПУБЛІКАЦІЇ

УДК 004.93
DOI:

С.М. ЛИСЕНКО, О.С. ВАНЯРХА, О.С. ВОРОБЙОВ
Хмельницький національний університет

МЕТОД ПОБУДОВИ АРХІТЕКТУРИ ВБУДОВАНИХ СИСТЕМ ОБРОБКИ ЗОБРАЖЕНЬ НА ОСНОВІ FPGA

У цій роботі розглянуто різні методи побудови архітектур на основі FPGA. Особлива увага приділяється моделям потоків даних та паралельним обчисленням, оскільки вони мають великий потенціал для розвитку адаптивних та ефективних апаратних архітектур. А також удосконалено метод побудови архітектури вбудованих систем обробки зображень, який на відміну від відомих використовує FPGA та грунтується на мінімізації потужності оптимізованому використанні пам'яті, і який забезпечує високу швидкодію обробки зображень.

Ключові слова: FPGA, інструменти високорівневого синтезу, цифрова обробка сигналів, вбудовані системи, системи обробки зображень, паралельні обчислення, архітектури пам'яті, високорівнева обробка зображень.

S. LYSENKO, O. VANIARKHA, O. VOROBIOV
Khmelnytskyi National University, Khmelnytskyi, Ukraine

A METHOD FOR CONSTRUCTING THE ARCHITECTURE OF EMBEDDED IMAGE PROCESSING SYSTEMS BASED ON FPGA

The paper presents a method for building the architecture of embedded image processing systems based on FPGA. The scientific novelty of this method is to improve the method of building the architecture of embedded image processing systems, which, unlike the known ones, uses FPGAs and is based on power minimization and optimized memory usage, and which provides high image processing performance. We describe the separation of image frames into BRAMs to maximize utilization (i.e., minimize the amount of memory required on the chip) while minimizing power consumption. The problem of utilization efficiency was formulated without paying any attention to power aspects. Power consumption was also integrated into the problem formulation. As a result, it was concluded that only one possible BRAM configuration is used for each image frame buffer. In this chapter, a model of the functioning of FPGA-based image processing software and hardware was developed and a method for building the architecture of embedded FPGA-based image processing systems was developed. The previous sections have explained that the memory allocation approach described is more efficient in terms of FPGA resource utilization. This is because reducing the memory usage allows for more logic to be implemented on the FPGA, which leads to better performance and faster data processing. The approach uses a more sophisticated addressing scheme, which enables different configurations of BRAM to control the memory, including signals for write enable and address decode. This increased complexity in memory management logic may result in a greater utilization of LUTs. However, studies have demonstrated that the additional LUT overhead is relatively insignificant compared to the positive impact that our approach has on performance and data processing speed.

Keywords: FPGA, high-level synthesis tools, digital signal processing, embedded systems, image processing systems, parallel computing, memory architectures, high-level image processing.

1 Вступ

Зміна технологічного ландшафту та швидка еволюція нових варіантів використання додатків надзвичайно важлива в теперішній час. Завдяки цьому стає очевидним, що потрібні адаптивні та ефективні апаратні архітектури, які здатні справлятися з різними видами робочих навантажень даних.

З метою вирішення цієї дослідницької проблеми необхідне вивчення різних методів побудови архітектур на основі FPGA. Особлива увага приділяється моделям потоків даних та паралельним обчисленням, оскільки вони мають великий потенціал для розвитку адаптивних та ефективних апаратних архітектур.

Для реалізації таких підходів необхідно здійснювати огляд передумов і пов'язаних з ними робіт. У зв'язку з цим, дана робота, містить у собі глибокий аналіз цих питань та дає можливість розглянути більш широкі питання, пов'язані з методами побудови архітектур на основі FPGA.

Актуальність роботи полягає в розробці методу побудови архітектури вбудованих систем обробки зображень на основі FPGA.

Метою роботи є підвищення швидкодії систем обробки зображень на основі FPGA.

Для підвищення швидкодії систем обробки зображень на основі FPGA необхідно:

1. дослідити методи побудови архітектури вбудованих систем обробки зображень на основі FPGA;
2. проаналізувати сучасні програмно-технічні засоби для обробки зображень на основі FPGA
3. розробити модель функціонування програмно-технічних засобів обробки зображень на основі FPGA;
4. розробити метод побудови архітектури вбудованих систем обробки зображень на основі FPGA;
5. реалізувати метод побудови архітектури вбудованих систем обробки зображень на основі FPGA.

Об'єкт дослідження – системи обробки зображень на основі FPGA.

Предмет дослідження – модель, метод побудови архітектури вбудованих систем обробки зображень на основі FPGA.

Наукова новизна отриманих результатів:

1. Удосконалено метод побудови архітектури вбудованих систем обробки зображень, який на відміну від відомих використовує FPGA та грунтується на мінімізації потужності оптимізованому використанні пам’яті, і який забезпечує високу швидкодію обробки зображень.
2. Набули подальшого розвитку програмно-технічні засоби обробки зображень на основі FPGA.

Відповідно, можна стверджувати, що вирішення задачі, яке проводиться в даній кваліфікаційній роботі, може мати значний вплив на розвиток адаптивних та ефективних апаратних архітектур вбудованих систем обробки зображень на основі FPGA, що здатні справлятися з обробкою динамічних робочих навантажень даних і в той же час забезпечувати швидкодію. Таким чином, дослідження проведені в роботі можуть внести значний внесок у розвиток комп'ютерних технологій, які використовуються в різних галузях.

2 Відомі методи побудови архітектури вбудованих систем обробки зображень на основі FPGA

Протягом останнього десятиліття багатопроцесорні архітектури стали важливою обчислювальною парадигмою для паралельних обчислень [1], [2], [4]. Вони стимулювали розвиток передових паралельних вбудованих архітектур. Тенденція до інтеграції однорідних і різнорідних обчислювальних блоків відкрила різні можливості апаратної декомпозиції, відображення та дослідження паралельних додатків. Апаратні архітектури, що складаються з десятків і сотень легких обчислювальних блоків, стали місцем не тільки для оптимізації продуктивності [2], [3], [5]. У той же час, ці апаратні архітектури створюють ряд проблем, таких як специфічні для архітектури навички перенесення та оптимізації додатків на базову архітектуру, що включає в себе управління та використання паралелізму та гетерогенності системи [5], [7]. Цей розділ охоплює базові дослідження, необхідні для розуміння цих проблем, а також підходи до побудови архітектури вбудованих систем обробки зображень на основі FPGA.

Нові гетерогенні багатопроцесорні системи на чіпові (MPSoC), такі як Xilinx Zynq-7000 і Altera Arria-V SoC, об'єднує можливість програмування програмного забезпечення процесора загального призначення (ARM) з апаратною програмованістю FPGA. Інтеграція апаратного та програмного забезпечення зробила

архітектуру MPSoC придатною обчислювальною платформою для реалізації змішаної функціональності на одному пристрої, а також для розробки адаптивних вбудованих архітектур [1], [4], [5].

Тим не менш, гетерогенні платформи MPSoC вирішили деякі проблеми апаратного та програмного забезпечення для програмування. Однак, пристосування паралельних обчислювальних задач до базових апаратних ресурсів шляхом використання більшої кількості обчислювальних вузлів, інтегрованих в один чіп, все ще залишається проблемою. Ця проблема безпосередньо пов'язана з оптимальним пошуком типу та ступеня паралелізму між декількома обчислювальними вузлами, доступними в гетерогенній системі. Крім того, оптимізована реалізація паралельних додатків з використанням цих гетерогенних платформ часто передбачає проектування та розробку апаратних прискорювачів для задоволення вимог додатків. Архітектура цих апаратних прискорювачів може мати динамічний діапазон гнучкості від фіксованої, реконфігурованої, програмно-програмованої або їх комбінації. Вони розміщуються на FPGA і зазвичай управляються процесором загального призначення, наприклад, процесором ARM Cortex-A [7].

Ринки універсальних додатків, що розвиваються, підвищують попит на високопродуктивні та ефективні архітектури на основі FPGA, які можуть обробляти робочі навантаження динамічних даних і в той же час адаптуються для прискорення різних програм. Один із способів підійти до цієї дослідницької проблеми полягає в розробці адаптивної апаратної архітектури на основі FPGA, яка дозволяє редагувати-компілювати-виконувати потік, знайомий розробникам програмного забезпечення та алгоритмів, замість синтезу апаратного забезпечення та розміщення та маршруту. Цього можна досягти шляхом заповнення логіки FPGA легкими та високопродуктивними програмними процесорами, які використовуються для програмованого апаратного прискорення. Ця базова архітектура буде адаптованою та може бути запрограмована за допомогою звичайних підходів до розробки програмного забезпечення, як показано на рисунку 1.1. Цей підхід не потребує синтезу дизайну апаратного забезпечення, місця та маршруту. Натомість йому знадобиться повторна компіляція програмного забезпечення, яке генеруватиме двійковий код для роботи на базових процесорах з програмним ядром.

Хоча проекти на основі HLS оптимізовані для варіантів використання, оскільки програма відома ще до реалізації основного апаратного забезпечення. Навпаки, у підході, що базується на процесорі, базова апаратна архітектура розроблена, синтезована, розміщено та маршрутизовано заздалегідь. Таким чином, очікується, що загальна площа буде більш значною, а продуктивність нижчою, ніж HLS, що відбуватиметься за рахунок адаптивності та скорочення часу розробки [8].

Цей підхід забезпечує апаратну абстракцію базових програмованих ресурсів FPGA, дозволяючи їм переконфігурувати за допомогою традиційних програмних підходів і відкриває це для розробника програмного забезпечення. Він успадковує переваги програмного забезпечення, такі як портативність, спільне проектування апаратно-програмного забезпечення, декомпозицію та параметри відображення для досягнення бажаної області та продуктивності. Крім того, уникнення необхідного ітераційного процесу синтезу та місця й маршруту зменшить час проектування, підвищить продуктивність і надасть можливість досліджувати проект, керований програмним забезпеченням. Джейн, Рігамонті та Лю повідомили про значне покращення шляхом компіляції додатків на архітектурі процесора через HDL та підходів часткової реконфігурації. Тим не менш, одним із важливих завдань є ефективна компіляція, відображення та виконання паралельних додатків на основну архітектуру програмованого апаратного прискорювача.

3 Основи методу побудови архітектури вбудованих систем обробки зображень на основі FPGA

Формалізувавши проблему використання, можна приступити до аналізу енергетичних наслідків кожної конфігурації.

Потрібно змоделювати динамічне енергоспоживання BRAM. Квант потужності споживається на читання та/або запис. Статична потужність BRAM прямо пропорційна використанню, тому розглядається в проблемі використання.

Для будь-якої комірки BRAM потужність зчитування споживається послідовністю операцій: тактовий сигнал стробується; прочитана адреса декодується; зчитані дані стробуються в колонковий мультиплексор; зчитані дані передаються на зовнішній порт BRAM.

Потужність запису споживається в такій послідовності: тактовий сигнал стробується; сигнал дозволу запису передає дані запису в буфери запису; рядок виділяється розшифровкою адреси; дані зберігаються в комірці RAM.

Далі розглядається поділ, де кожна дана розподілена між вісьмома BRAM, і поділ, де кожна дана розподілена між двома BRAM.

Кожна операція читання/запису в першому повинна споживати енергію в чотири рази більше кількості BRAM в останньому. На рисунках 3.1 і 3.2 зображено приклади споживання електроенергії для двох схем розподілу.

Схема поділу, яка мінімізує горизонтальне використання BRAM (тобто через x), більше підходить для стробування тактового сигналу.

Оскільки за одну операцію необхідно отримати доступ до меншої кількості блоків BRAM, пропорція невикористаних блоків, які можуть бути ефективно закриті, збільшується.

Це просто реалізувати тактовий строб за допомогою вибору дозволу мікросхеми, який увімкнено/вимкнено на основі декодування адреси. colorred Іншими словами, енергоспоживання BRAM пропорційне кількості BRAM, необхідних для доступу до кожного пікселя: і це число залежить від обраної конфігурації.

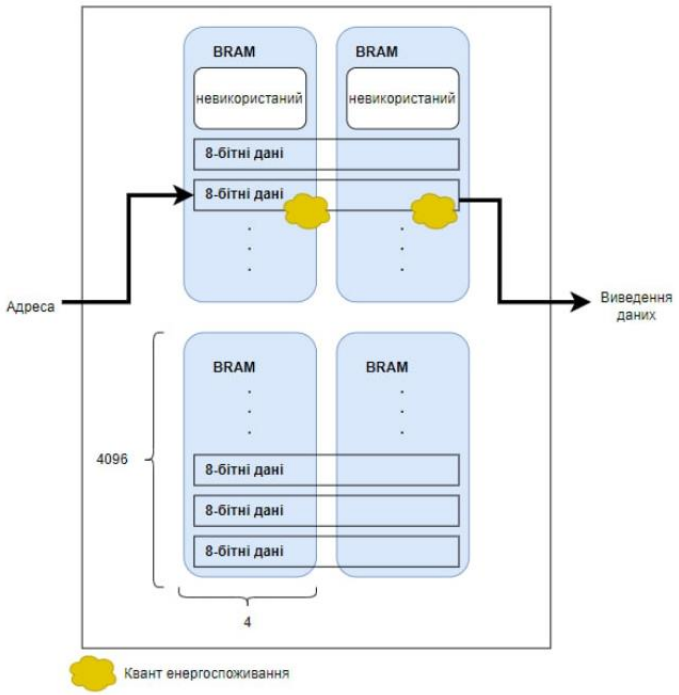


Рисунок 1 - Розбиття на дві горизонтальні частини між двома блоками запам'ятовування з випадковим доступом (BRAM). Кожен доступ використовує дві одиниці кількості споживання енергії

Інтуїтивно зрозумілий підхід до балансування енергоспоживання та використання полягає в тому, щоб завжди використовувати найширшу конфігурацію BRAM, достатньої для Bw, або кратні найширшій доступній. Однак це не оптимізована стратегія. Хоча це правда, що динамічна потужність зменшується, статична потужність може збільшуватися при переході від однієї конфігурації до ширшої, оскільки загальна кількість BRAM може збільшитися: ефективність використання змінюється. Крім того, логіка, необхідна для

сигналів адреси (і мікросхеми), збільшується при переході до ширшої конфігурації. Цей аспект робить проблеми використання та живлення неподільними. Таким чином необхідне балансування цих двох аспектів.

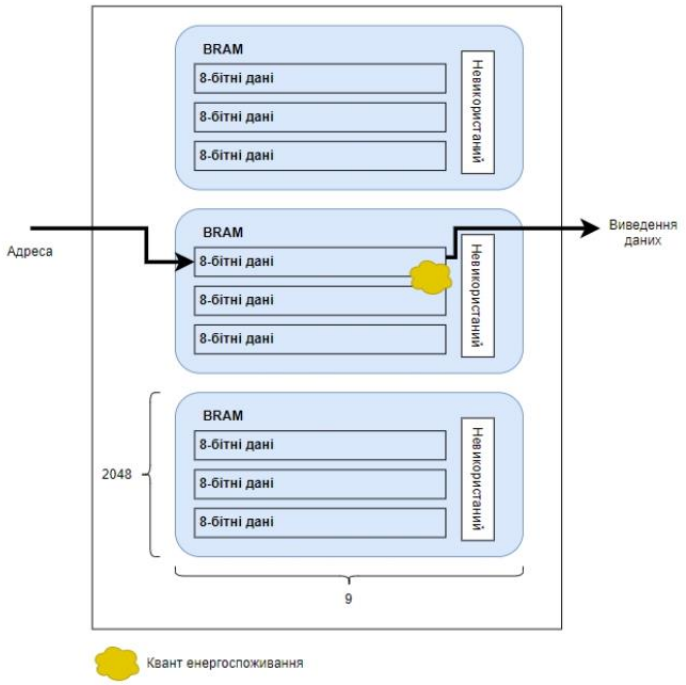


Рисунок 2 - Розбиття на горизонтальні частини в межах одного блоку запам'ятовування з випадковим доступом (BRAM). Кожен доступ використовує одну одиницю кількості споживання енергії

3.1 Розбиття для живлення та використання

Для кожного елемента в наборі конфігурацій Cfg (що має загалом і елементів) процедура обчислює необхідну кількість BRAM для зберігання кадру шириною W, висотою H і шириною Bw, ефективність такої конфігурації та порівнює її з найвища ефективність, знайдена на даний момент.

Тут увага зосереджена виключно на використанні. По суті, це вичерпний пошук, оскільки кількість можливих конфігурацій пам’яті обмежена, і це автономний процес.

У таблиці 1 зображено конфігурації, вибрані процедурою для репрезентативної кількості розмірів кадру та бітової ширини пікселів. Деякі з конфігурацій не оптимізовані за енергоспоживанням: зауважте, що для пікселів шириною 10, 14 і 22 найчастіше вибирається конфігурація BRAM 2 × 8192 (споживає потужність на 5, 7 і 11 BRAM на доступ відповідно). Це інтуїтивно зрозуміло з точки зору ефективності використання: це єдина конфігурація, яка розділяє ширину, і відповідає вибору конфігурації 4 × 4096 для пікселів шириною 8, 12, 20 і 24 і конфігурації 18 × 1024 для пікселів ширини. ширина 18.

Ця нелінійність ускладнює розробку оптимізованої процедури розподілу як за використанням, так і за енергоефективністю.

Таким чином, можна використовувати більш спокійний підхід і визначити процедуру через визначені користувачем компроміси (тобто оцінку того, скільки використання BRAM можна обміняти на зменшення потужності) і евристику потужності та простору на основі емпіричних властивостей. Передбачається, що компроміс виражається у відсоткових пунктах.

Процедура починається з вибору оптимізованого рішення щодо використання та повторення ширших конфігурацій BRAM (у вимірі x), обчислення ефективності використання.

Таблиця 1 - Конфігурації BRAM на основі оптимізованої процедури використання.

Ширина пікселів									
Кадр	8	10	12	14	16	18	20	22	24
160 x 120	4 x 4096	4 x 4096	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	4 x 4096
320 x 240	4 x 4096	2 x 8192	4 x 4096	2 x 8192	18 x 1024	18 x 1024	4 x 4096	2 x 8192	4 x 4096
512 x 512	4 x 4096	2 x 8192	4 x 4096	2 x 8192	4 x 4096	18 x 1024	4 x 4096	2 x 8192	1 x 16384
640 x 480	4 x 4096	2 x 8192	4 x 4096	2 x 8192	18 x 1024	18 x 1024	4 x 4096	2 x 8192	4 x 4096
1280 x 720	4 x 4096	2 x 8192	4 x 4096	2 x 8192	18 x 1024	18 x 1024	4 x 4096	2 x 8192	4 x 4096

Поки рівень використання перевищує порогове значення, визначене різницею між найкращим використанням і компромісом у відсоткових пунктах, процедура продовжується. Коли він знаходить перше рішення нижче порогового значення, він виходить, повертаючи останнє рішення вище порогового значення.

Цей підхід відповідає евристиці моделі потужності, описаній у раніше: енергоспоживання зменшується зі збільшенням горизонтальної ширини BRAM, див. рис. 3 і 4.

У таблиці 2 зображено конфігурації BRAM, вибрані за збалансованою процедурою, із встановленим компромісом у 12 відсоткових пунктів. Порівняно з оптимізованими конфігураціями, більшість ширини збільшено, що призводить до більш енергоефективного рішення на основі вищезгаданої евристики.

Таблиця 2 - Конфігурації BRAM на основі збалансованої процедури з компромісом, рівним дванадцяти відсотковим пунктам

Ширина пікселів									
Кадр	8	10	12	14	16	18	20	22	24
160 x 120	9 x 2048	4 x 4096	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048
320 x 240	9 x 2048	4 x 4096	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048
512 x 512	9 x 2048	2 x 8192	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048
640 x 480	9 x 2048	2 x 8192	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048
1280 x 720	9 x 2048	2 x 8192	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048

3.1 Застосування розділення пам'яті

Вищеописані алгоритми можна використовувати як у потоках проектування HDL, так і в потоках проектування HLS: у потоці проектування HDL, керуючи впровадженням та/або рефакторингом дизайнера; у потоці проектування HLS через інтеграцію в підсистему генерації коду інструментів синтезу.

На рисунку 3.3 зображено запропоновані проектні потоки. Додаткові кроки можна виконувати вручну, або починаючи з проектів HDL, або змінюючи вихідні дані HLS перед синтезом; за допомогою автоматизованих засобів рефакторингу, які обчислюють запропоновані процедури; або інструментом HLS до генерації коду. Описується ручний процес, який використовується в наших експериментах.

Після того, як структуру пам’яті було отримано зі специфікації процедури, відповідно до рівнянь та алгоритмів, обчислюються для визначення розподілу BRAM. BRAM обчисленої конфігурації створюється та міститься в модулях (тобто апаратних об’єктах).

Верхній модуль створює екземпляри всіх підмодулів, надаючи інтерфейси, ідентичні базовому дизайну HDL або специфікації інструменту HLS.

Логіка адресації всередині верхнього модуля керує мікросхемою вмикання сигналів для кожного підмодуля, гарантуючи, що неадресовані BRAM не ввімкнено.

Цей ретельний розподіл логіки HDL в ієрархічних модулях, де логіка адресації визначається міжз’єднанням верхнього рівня, а конфігурація BRAM визначається параметрами конфігурації модуля, забезпечує використання бажаних конфігурацій (це базується на наших експериментах із використанням Vivado: різні інструменти синтезу можуть вимагають додаткових прагм компілятора).

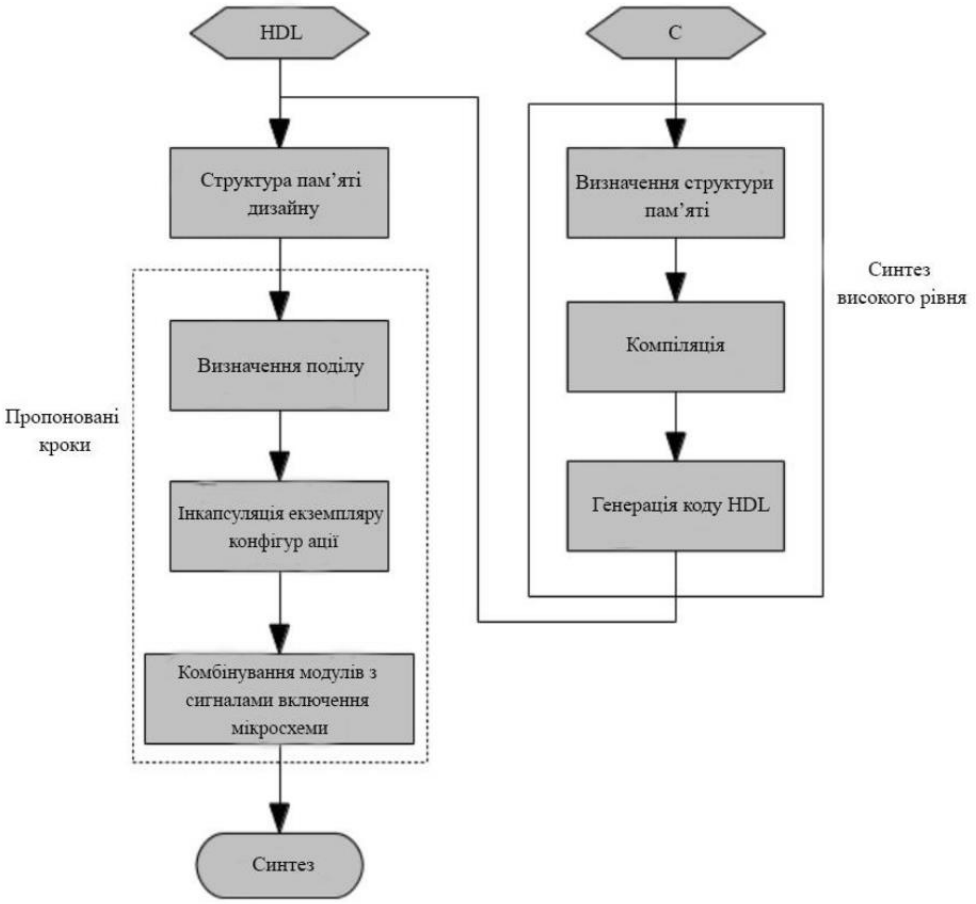


Рисунок 3 - Запропонований проектний потік від HDL і HLS, підкреслюючи додаткові кроки, необхідні для мінімізації використання та потужності

Висновки

В цій роботі було представлено метод побудови архітектури вбудованих систем обробки зображень на основі FPGA. Наукова новизна цього методу полягає в удосконаленні методу побудови архітектури вбудованих систем обробки зображень, який на відміну від відомих використовує FPGA та ґрунтується на мінімізації потужності, та оптимізованому використанні пам'яті, і який забезпечує високу швидкодію обробки зображень. Був проведений опис розділення кадрів зображень на BRAM, щоб максимізувати використання (тобто мінімізувати кількість необхідної пам'яті на чіпі) за умови мінімізації споживання енергії. Було сформульовано проблему ефективності використання, не звертаючи жодної уваги на аспекти потужності. Також було інтегровано енергоспоживання в формулювання проблеми. В результаті було здійснено до висновку, що для кожного кадрового буфера зображення використовується лише одна можлива конфігурація BRAM. В даному розділі було розроблено модель функціонування програмно-технічних засобів обробки зображень на основі FPGA та розроблено метод побудови архітектури вбудованих систем обробки зображень на основі FPGA.

Література

1. M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos. Edge Analytics in the Internet of Things. *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 24 – 31, 2015.

A. Gandomi and M. Haider. Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management(IJIM)*, vol. 35, no. 2, pp. 137 – 144, 2015.

2. Video Analytics Hardware, Software, and Services Revenue to Reach $3 Billion by 2022. URL: https://www.embedded-vision.com/industry-analysis/market-analysis/ video-analytics-hardware-software-and-services-revenue-reach-3-bil (дата доступу 10.05.2023)

3. L. Markov, "Limits on Fundamental Limits to Computation," Coputing Research Laboratory (CoRR), vol. abs/1408.3821, 2014. Available: http://arxiv.org/abs/1408.3821

4. Programmable embedded platforms for remote and compute intensive image processing applications (EP/K009583/1)." URL: Available: http://gow.epsrc.ac.uk/NGBOViewGrant.aspx?GrantRef=EP/K009583/1

5. R. Stewart, K. Duncan, G. Michaelson, P. Garcia, D. Bhowmik, and A.Wallace, "RIPL: A Parallel Image Processing Language for FPGAs," ACM Transactions on Reconfigurable Technology and Systems (TRETS, vol. 11, no. 1, Mar. 2018.

6. E. Bezati, "High-level synthesis of dataflow programs for heterogeneous platforms," Ph.D. dissertation, EPFL, Lausanne, Switzerland, 2015. Available: https://infoscience.epfl.ch/record/207992/files/EPFLTH6653.pdf

7. D. F. Bacon, R. Rabbah, and S. Shukla, "FPGA Programming for the asses," ACM Queue Magazine, vol. 11, no. 2, pp. 40 – 52, Feb. 2013.

8. M. Gort and J. Anderson, "Design re-use for compile time reduction in FPGA high-level synthesis flows," in Proc. IEEE International Conference on Field-Programmable Technology (FPT), Shanghai, China, Dec. 2014, pp. 4 – 11.

9. G. Stitt and J. Coole. Intermediate Fabrics: Virtual Architectures for Near-Instant FPGA Compilation. *IEEE Embedded Systems Letters*, vol. 3, no. 3, pp. 81 – 84, 2011.

10. R. Baxter, S. Booth, M. Bull, G. Cawood, K. D'Mellow, X. Guo, M. Parsons, J. Perry, A. Simpson, and A. Trew, High-Performance Reconfigurable Computing - the View from Edinburgh. *Proceedings of the Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Washington, USA, 2007, pp. 373 – 279.

11. H. Chenini, J. P. D´erutin, R. Aufr`ere, and R. Chapuis, Parallel embedded processor architecture for FPGA-based image processing using parallel software skeletons. *EURASIP Journal on Advances in Signal Processing*, vol. 2013, no. 1, p. 153, 2013.

12. L. Natvig, A. Iordan, M. Eleyat, M. Jahre, and J. Amundsen, Multi- and Many-Cores, Architectural Overview for Programmers, 2017, ch. 1, pp. 1 –27.

A. Vajda, Multi-core and Many-core Processor Architectures, Boston, USA, 2011, ch. 2, pp. 9 – 43.

13. Andryc, T. Thomas, and R. Tessier. Soft GPGPUs for Embedded FPGAs: An Architectural Evaluation. CoRR, vol. abs/1606.06454, 2016. Available: http://arxiv.org/abs/1606.06454

14. N. Kapre, "Custom FPGA-based soft-processors for sparse graph acceleration," in Proc. 26th IEEE International Conference on Applicationspecific Systems, Architectures and Processors (ASAP), Toronto, Canada, Jul. 2015, pp. 9 – 16.

15. T. Lieske, M. Reichenbach, B.Ringlein, and D. Fey, "Dataflow optimization for programmable embedded image preprocessing accelerators," in Proc. IEEE International Conference on ReConFigurable Computing and FPGAs (ReConFig), Cancun, Mexico, Nov. 2016, pp. 1 – 8.

16. M. Schoeberl, S. Abbaspour, B. Akesson, N. Audsley, R. Capasso, J. Garside, K. Goossens, S. Goossens, S. Hansen, R. Heckmann, S. Hepp, B. Huber, A. Jordan, E. Kasapaki, J. Knoop, Y. Li, D. Prokesch, W. Puffitsch,187

17. P. Puschner, A. Rocha, C. Silva, J. Spars, and A. Tocchi, "T-CREST: Time-predictable multi-core architecture for embedded systems," Journalof Systems Architecture (JSA), vol. 61, no. 9, pp. 449 – 471, 2015.

18. Heisswolf, A. Zaib, A. Weichslgartner, M. Karle, M. Singh, T. Wild, J. Teich, A. Herkersdorf, and J. Becker, "The invasive network on chip - a multi-objective many-core communication infrastructure," in Proc. IEEE International Workshop on Architecture of Computing Systems (ARCS), Luebeck, Germany, Feb. 2014, pp. 1 – 8.

19. D. She, Y. He, L. Waeijen, and H. Corporaal, "A Co-Design Framework with OpenCL Support for Low-Energy Wide SIMD Processor," Journal of Signal Processing Systems (JSPS), vol. 80, no. 1, pp. 87 – 101, Jul. 2015.

20. S. Fernando, F. Siyoum, Y. He, A. Kumar, and H. Corporaal, "MAMPSx: A design framework for rapid synthesis of predictable heterogeneous MPSoCs," in Proc. IEEE International Symposium on Rapid System Prototyping (RSP), Montreal, Canada, Oct. 2013, pp. 136 – 142.

21. J. S´erot, F. Berry, and C. Bourrasset, "High-level dataflow programming for real-time image processing on smart cameras," Journal of Real-Time Image Processing (JRTIP, vol. 12, no. 4, pp. 635 – 647, 2016.

22. C. Liu, H. C. Ng, and H. K. H. So, "QuickDough: A rapid FPGA loop accelerator design framework using soft CGRA overlay," in Proc. IEEE International Conference on Field Programmable Technology (FPT), Queenstown, New Zealand, Dec. 2015, pp. 56 – 63. 188

ДОДАТОК Б
(обов’язковий)

ПРЕЗЕНТАЦІЯ

МЕТОД ПОБУДОВИ АРХІТЕКТУРИ ВБУДОВАНИХ СИСТЕМ ОБРОБКИ ЗОБРАЖЕНЬ НА ОСНОВІ FPGA

Виконав студент групи КІ2м-21-1 Ванярха О.С.
Науковий керівник д.т.н., професор Лисенко С.М.

Об’єкт, предмет та мета дослідження

Метою роботи є підвищення швидкодії систем обробки зображень на основі FPGA.

Об'єктом дослідження є системи обробки зображень на основі FPGA.

Предметом дослідження є модель, метод побудови архітектури вбудованих систем обробки зображень на основі FPGA

Наукова новизна

- удосконалено метод побудови архітектури вбудованих систем обробки зображень, який на відміну від відомих використовує FPGA та грунтується на мінімізації потужності, та оптимізованому використанні пам'яті, і який забезпечує високу швидкодію обробки зображень;
- набули подальшого розвитку програмно-технічні засоби обробки зображень на основі FPGA.

Актуальність роботи

- Зображення є одним з основних джерел інформації в сучасному світі, і обробка зображень відіграє важливу роль в багатьох областях, таких як медицина, наука про матеріали, технології виробництва, автомобільна промисловість, безпека і багато інших. FPGA забезпечують швидку та ефективну обробку зображень, що є дуже важливим для багатьох застосувань, де час обробки є критичним фактором.
- Також обробка зображень є дуже складним процесом, і потребує значних обчислювальних ресурсів. FPGA дозволяють виконувати обчислення паралельно, що забезпечує більш швидку та ефективну обробку зображень порівняно з традиційними процесорами. Крім того, FPGA можна програмувати та перепрограмовувати, що дозволяє забезпечувати гнучку та адаптивну обробку зображень залежно від конкретних потреб застосування.

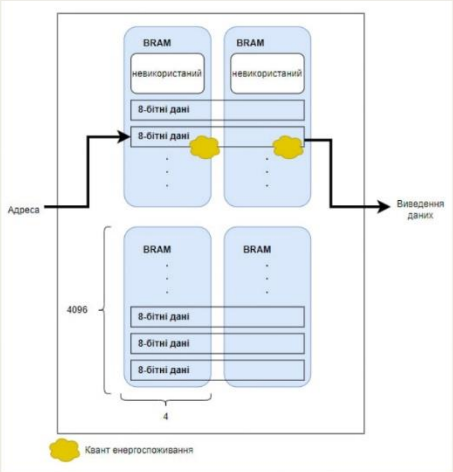
В ході роботи використовується термін BRAM. це поняття є досить поширеним в контексті FPGA, і відповідає блоковій пам'яті з довільним доступом.

За допомогою BRAM можна зберігати дані, що використовуються у процесі обчислень, що є дуже важливим для багатьох застосувань FPGA.

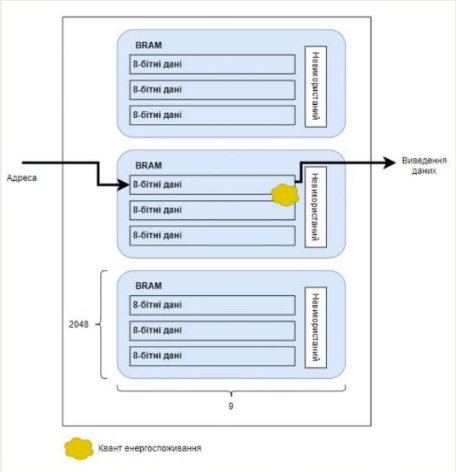
Остаточні результати дослідження, які описані в роботі, мають великий потенціал для практичного використання в різних сферах, таких як обробка зображень, відео аналітика, машинне навчання.

Рішення полягає у використанні стратегії динамічного управління пам'яттю, такої як кешування.

Це дозволяє зберігати лише необхідні дані в обмеженому обсязі пам'яті, що підвищує ефективність виконання глобальних операцій та зменшує витрати на обробку даних.



Розбиття на дві горизонтальні частини між двома BRAM



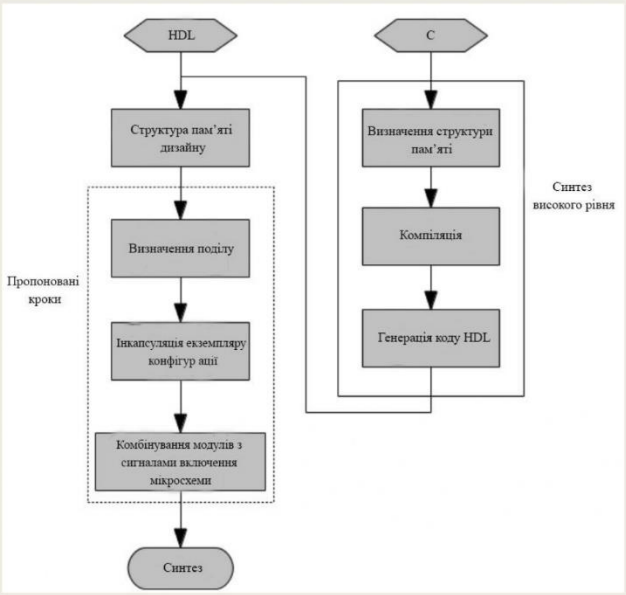
Розбиття на горизонтальні частини в межах одного BRAM

Кадр	Ширина пікселів								
	8	10	12	14	16	18	20	22	24
160 x 120	4 x 4096	4 x 4096	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	4 x 4096
320 x 240	4 x 4096	2 x 8192	4 x 4096	2 x 8192	18 x 1024	18 x 1024	4 x 4096	2 x 8192	4 x 4096
512 x 512	4 x 4096	2 x 8192	4 x 4096	2 x 8192	4 x 4096	18 x 1024	4 x 4096	2 x 8192	1 x 16384
640 x 480	4 x 4096	2 x 8192	4 x 4096	2 x 8192	18 x 1024	18 x 1024	4 x 4096	2 x 8192	4 x 4096
1280 x 720	4 x 4096	2 x 8192	4 x 4096	2 x 8192	18 x 1024	18 x 1024	4 x 4096	2 x 8192	4 x 4096

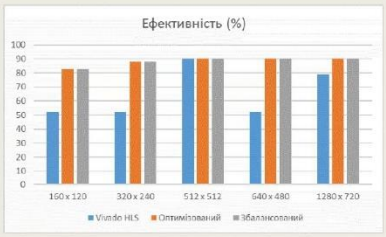
Конфігурації BRAM на основі оптимізованої процедури використання

Кадр	Ширина пікселів								
	8	10	12	14	16	18	20	22	24
160 x 120	9 x 2048	4 x 4096	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048
320 x 240	9 x 2048	4 x 4096	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048
512 x 512	9 x 2048	2 x 8192	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048
640 x 480	9 x 2048	2 x 8192	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048
1280 x 720	9 x 2048	2 x 8192	4 x 4096	18 x 1024	18 x 1024	18 x 1024	4 x 4096	9 x 2048	9 x 2048

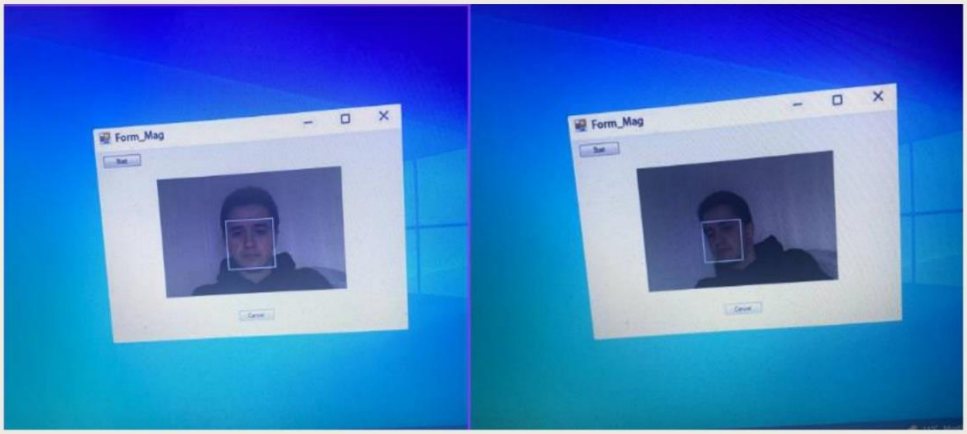
Конфігурації BRAM на основі збалансованої процедури використання



Запропонований проектний потік від HDL і HLS



Гістограми ефективності



Процес відстеження обличчя в реальному часі,
що відображається на ПК

Висновки

Ефективне зіставлення високорівневих описів кадрів зображень із низькорівневими системами пам’яті є важливим фактором для широкого впровадження FPGA як платформ для розгортання програм обробки зображень високого рівня. Алгоритми розділення є одним із методів проектування, який забезпечує шляхи до енергоефективних проектів, які можуть відповідати вимогам сучасних додатків.

В роботі були розглянуті питання, пов'язані з підходами до проектування апаратного забезпечення на основі FPGA, моделями потоків даних і паралельними обчисленнями, а також дається огляд їх передумов і пов'язаних з ними робіт.

Дякую за увагу

UNICHECK

Ім'я користувача:
Кафедра КІ

ID перевірки:
1015000834

Дата перевірки:
09.05.2023 19:38:42 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
09.05.2023 19:39:01 EEST

ID користувача:
100005591

Назва документа: Ванярха_Метод побудови архітектури вбудованих систем обробки зображень на основі FP...

Кількість сторінок: 85 Кількість слів: 16605 Кількість символів: 125859 Розмір файлу: 2.25 MB ID файлу: 1014691531

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

3.97%
Схожість

Найбільша схожість: 0.64% з Інтернет-джерелом (https://christuniversity.in/academic-publication/BANGALORE%20KENG.

3.44% Джерела з Інтернету | 497 ..... Сторінка 87

1.26% Джерела з Бібліотеки | 69 ..... Сторінка 90

1.48% Цитат

Цитати | 6 ..... Сторінка 91

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 34

Підозріле форматування | 13 сторінок

Tue May 09 18:45:45 EEST 2023, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 28.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 13%**

	Документ		Сумарний збіг по Базі Даних	
ID: 113152 Назва: МКР Метод побудови архітектури вбудованих систем обробки зображень на основі FPGA Додано в БД: 2023-05-09 Автора: Ванярха О.С. Керівники: Лисенко С.М. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	106420	738	29660 (28%)	232 (31%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
112080	Назва: ЗВІТ з науково-дослідної практики "Метод побудови архітектури вбудованих систем обробки зображень на основі FPGA" Додано в БД: 2023-03-21 Автора: Ванярха О.С. Керівники: Бобровнікова К.Ю. Консультанти: Опоненти:	29427 (28.0%)	231 (31.0%)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач: Ванярха Олександр Сергійович

Тема: Метод побудови архітектури вбудованих систем обробки зображень на основі FPGA

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень – ; кількість сторінок записки 70

1. Короткий зміст роботи та прийнятих рішень У роботі запропоновано метод побудови архітектури вбудованих систем обробки зображень на основі FPGA

2. Висновок про відповідність роботи дипломному завданню Кваліфікаційна робота магістра відповідає виданому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проведено огляд систем обробки зображень на основі FPGA. Досліджено відомі рішення та засоби в цій сфері. У другому розділі запропоновано модель вбудованої системи обробки зображень на основі FPGA. У третьому розділі запропоновано метод побудови архітектури вбудованих систем обробки зображень на основі FPGA. У четвертому розділі запропоновано реалізацію методу побудови архітектури вбудованих систем обробки зображень на основі FPGA.

4. Позитивні сторони роботи: Запропонований метод побудови архітектури вбудованих систем обробки зображень, на відміну від відомих використовує FPGA та грунтується на мінімізації потужності, та оптимізованому використанні пам'яті, і який забезпечує високу швидкодію обробки зображень. Негативні сторони роботи: В роботі присутні певні логічні помилки щодо опису моделей побудови архітектури вбудованих систем обробки зображень на основі FPGA.

6. Оцінка графічного оформлення та пояснювальної записки роботи: –

7. Відгук про роботу в цілому: В загальному робота виконана на невисокому рівні.

8. Інші зауваження: –

9. Оцінка кваліфікаційної роботи:

Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи вважаю, що робота заслуговує оцінки «задовільно» 3.00 (E)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) д.т.н.. професор, Мартинюк В.В., завідувач кафедри

"10 травня" 2023р.

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Вакуріна Олександра Сергійовича
ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-21-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіатоповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

10.05.2023
дата

підпис

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Метод побудови архітектури вбудованих систем обробки зображень на основі FPGA

Автор: Ванярха Олександр Сергійович

Спеціальність: 123 – Компютерна інженерія

Освітня програма: освітньо-наукова

Науковий керівник: Лисенко Сергій Миколайович, д.т.н, професор

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах є збіг зі звітом з науково-дослідної практики автора Ванярхи Олександра " Метод побудови архітектури вбудованих систем обробки зображень на основі FPGA ", який було додано в репозитраій ХНУ 21 березня 2023 року;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;
4) в якості запозичень в окремих місцях системою зафіксовано послідовності формул та кодів, які є вхідними даними до великої кількості задач і не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;
5) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості Unicheck, складає 3.97% і адресується до 566 першоджерела; та системою Anti-Plagiarism складає 28%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ С.М. Лисенко

Гарант ОП _____ О. С. Савенко

Завідувач кафедри КІІС _____ Т. О. Говорущенко